The Transactor

The Tech/News Journal For Commodore Computers Vol. 4

Issue 05
$2.95

Special Reference Issue

Everything For Plugging Into Your Commodore Equipment

And All In One, Handy Place! Contains:

- BASIC and Machine Language Summaries
- Screen, Sprite, and Character Design Grids
- New Butterfield SuperCharts
- Memory Maps For All Machines
- Disk Specs For All Drive Types
- Port Pinouts For All Machines
- Colour and Sound Tables
- Book List and Glossary
- ROM Entry Points
- Plus LOADs More!

B.W. BROOKS
RR 3
TRENTON, ONT
CANADA K8V 5P6

0 57913 70974 05

Richvale Telecommunications

10610 BAYVIEW (Bayview Plaza)
RICHMOND HILL, ONTARIO, CANADA L4C 3N8
(416) 884-4165

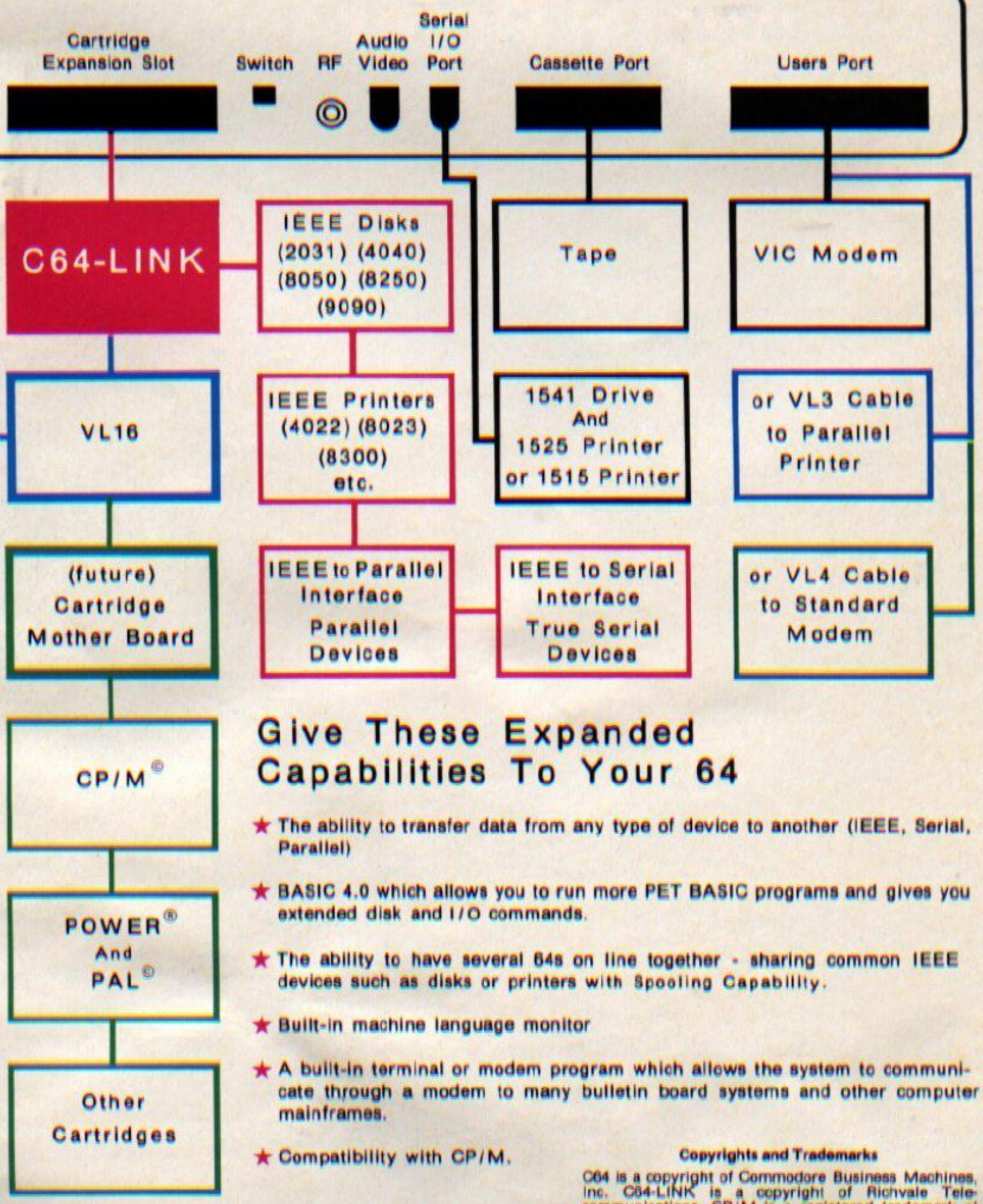
$185 Canadian

C64-LINK©

The Smart 64

Also available for VIC 20

RTC RTC



Give These Expanded Capabilities To Your 64

★ The ability to transfer data from any type of device to another (IEEE, Serial, Parallel)

★ BASIC 4.0 which allows you to run more PET BASIC programs and gives you extended disk and I/O commands.

★ The ability to have several 64s on line together - sharing common IEEE devices such as disks or printers with Spooling Capability.

★ Built-in machine language monitor

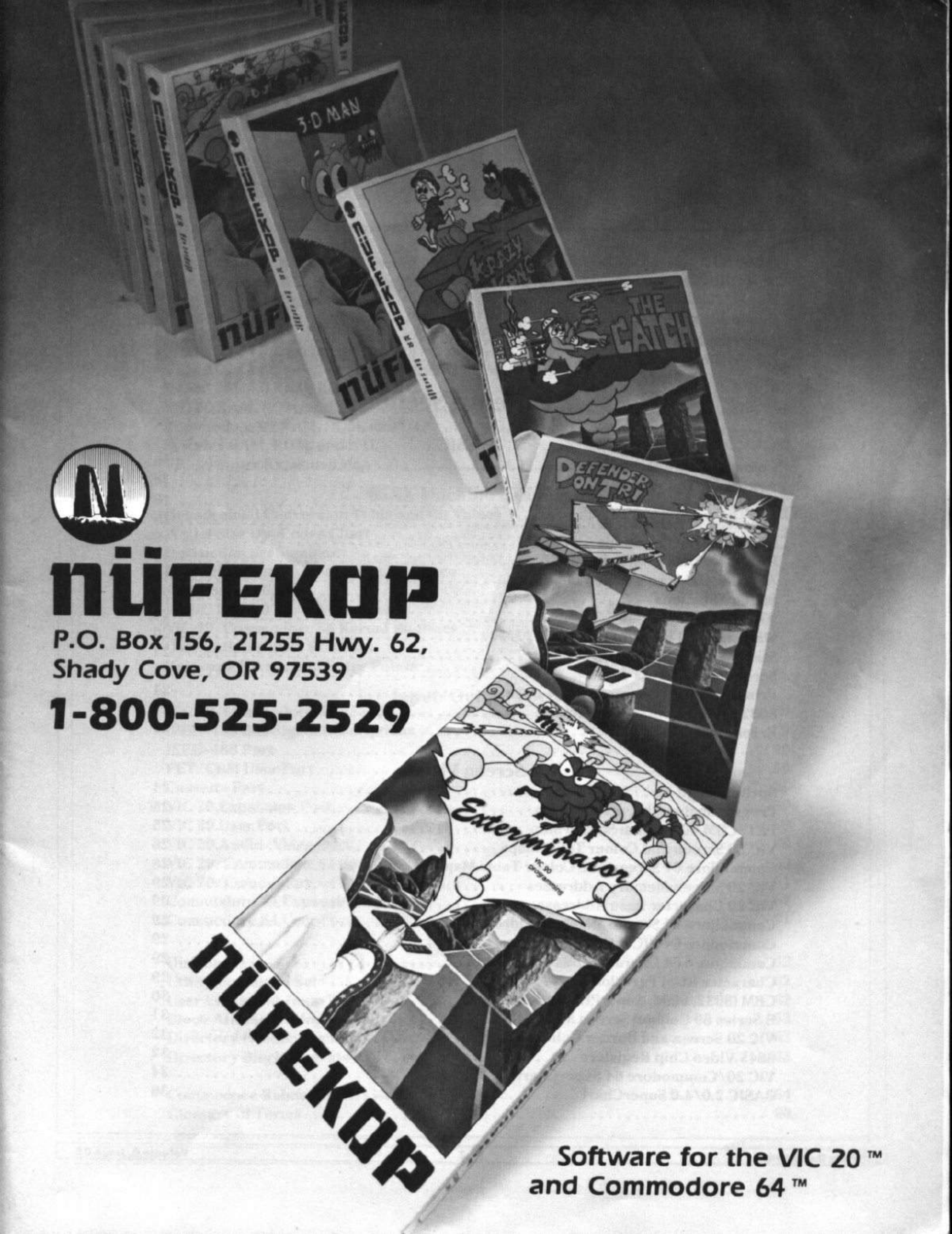
★ A built-in terminal or modem program which allows the system to communicate through a modem to many bulletin board systems and other computer mainframes.

★ Compatibility with CP/M.

Contact your local Commodore dealer or RTC.

Copyrights and Trademarks

C64 is a copyright of Commodore Business Machines, Inc. C64-LINK is a copyright of Richvale Telecommunications. CP/M is a registered trademark of Digital Research. POWER is a trademark of Professional Software. PAL is a copyright of Brad Templeton.



nüFEKOP

P.O. Box 156, 21255 Hwy. 62,
Shady Cove, OR 97539

1-800-525-2529

Software for the VIC 20™ and Commodore 64™

The Transactor

Editorial 5
News BRK 6
The MANAGER Column 10
CompuKinks 15
Advertising Section 70
Advertising Index 88

Reference Section

BASIC

Commands and Statements 16
Arithmetic Operators 18
Special Symbols 18
Reserved Variables 18
Hierarchy of Operations 18
BASIC 4.0 Disk Commands 19
Additional B Series Commands 19
String Functions 20
Arithmetic Functions 20
Mathematical Functions 20
Status Variables – ST, DS & DS$ 21
Secondary Address Table 21
Printer Control Characters 21
8032 Screen Control Characters 21
Error Messages 22

Screen Visual

Sprite Design 24
Programmable Character Design 25
PET/CBM 40 Column Screen Map 25
VIC 20 Screen and Colour Table Maps 26
Commodore 64 Screen and Colour Table Maps 28
VIC 20 Screen Memory Addresses 29
VIC 20 Character Base Addresses 29
Commodore 64 Screen Memory Addresses 29
Commodore 64 VIC II Chip Addresses 29
Commodore 64 Character Base Addresses 29
Character ROM Partitioning 29
CBM (8032, 8096, SuperPET) Screen Map 30
B Series 80 Column Screen Map 31
VIC 20 Screen and Border Colours 32
6845 Video Chip Registers 32
VIC 20/Commodore 64 SuperChart 34
BASIC 2.0/4.0 SuperChart 36

Volume 4, Issue 05
The Reference Issue

Sound

Commodore 64 SID Note Values 33
Commodore 64 ADSR Envelope Values 33
VIC 20 Note Values 33

Memory Maps

BASIC 2.0/4.0 Zero Page Contents at Power–Up 38
VIC 20/Commodore Zero Page Contents at Power–Up 40
BASIC 2.0/4.0 RAM, ROM, and I/O 42
VIC 20 RAM, ROM, and I/O 44
Commodore 64 RAM, ROM, and I/O 46
B Series RAM, ROM, and I/O 48
VIC 20 Super Expander Map 50

65XX Machine Language

Hexadecimal Conversion Table and Bit Values 32
Wallet Size Op–Codes Chart 32
Instruction Set Summary 52
Instruction Set – Alphabetical Sequence 54
Addressing Modes 54
BASIC 2.0/4.0 Kernal Routines 55
VIC 20/Commodore 64 Kernal Routines 55
User Callable ROM Routines 56
Keyword Tokens and Entry Points 57

Input/Output Ports

VIC 20/Commodore 64 RS–232 58
IEEE–488 Bus Signal Descriptions 59
IEEE–488 Port 59
PET/CBM User Port 59
Cassette Port 59
VIC 20 Expansion Port 60
VIC 20 User Port 60
VIC 20 Audio/Video Port 60
VIC 20/Commodore 64 Serial Port 60
VIC 20/Commodore 64 Joystick Ports 61
Commodore 64 Expansion Port 61
Commodore 64 User Port 61

Disk Drives

Disk Specs 62
Utility Command Set 62
User Command Jump Table 62
Block Allocation Map Formats 63
Directory Header Formats 63
Directory Block Formats 63

Commodore Related Book List 64
Glossary of Terms 66

The Transactor
The Tech/News Journal For Commodore Computers

Managing Editor
Karl J. H. Hildon

Editor
Richard Evers

Advertising Manager
Kelly M. George
416 826 1662

Art Director
John Mostacci

Contributing Writers
Greg Beaumont
Don Bell
Dave Berezowski
Jim Butterfield
Gord Campbell
Chuan Chee
Mike Donegan
Alex Gaul
Donna Green
Fred Hambrecht
Paul Higginbottom
Dave Hook
Eike Kaiser
Peter Lear
Bill MacLean
Darren Spruyt
John Stoveken
Brad Templeton
Don White

Production
Attic Typesetting Ltd.

Printing
Printed in Canada by
MacLean Hunter Printing

Program Listings In The Transactor

All programs listed in The Transactor will appear as they would on your screen in Upper/Lower case mode. To clarify two potential character mix-ups, zeroes will appear as '0' and the letter "o" will of course be in lower case. Secondly, the lower case L ('l') has a flat top as opposed to the number 1 which has an angled top.

Many programs will contain reverse video characters that represent cursor movements, colours, or function keys. These will also be shown exactly as they would appear on your screen, but they're listed here for reference.

Occasionally programs will contain lines that show consecutive spaces. Often the number of spaces you insert will not be critical to correct operation of the program. When it is, the required number of spaces will be shown. For example:

print" flush right" – would be shown as – print"[space10]flush right"

Cursor Characters For PET / CBM / VIC / 64

Down – [q]
Up – [Q]
Right – [ ]
Left – [Lft]
RVS – [r]
RVS Off – [R]
Insert – [T]
Delete – [t]
Clear Scrn – [S]
Home – [s]
STOP – [c]

Colour Characters For VIC / 64

Black – [P]
White – [e]
Red – [£]
Cyan – [Cyn]
Purple – [Pur]
Green – [↑]
Blue – [←]
Yellow – [Yel]
Orange – [A]
Brown – [U]
Lt. Red – [V]
Grey 1 – [W]
Grey 2 – [X]
Lt. Green – [Y]
Lt. Blue – [Z]
Grey 3 – [Gr3]

Function Keys For VIC / 64

F1 – [E]
F2 – [I]
F3 – [F]
F4 – [J]
F5 – [G]
F6 – [K]
F7 – [H]
F8 – [L]

The Transactor is published quarterly by Transactor Publishing Inc. It is in no way connected with Commodore Business Machines Ltd. or Commodore Incorporated. Commodore and Commodore product names (PET, CBM, VIC, 64) are registered trademarks of Commodore Inc.

Volume 4 Subscriptions: Canada $15 Cdn
U.S.A. $15 US.
All other $18 US.

Second Class Mail
Permit Pending

Back issues are still available for Volume 3: $15 Cdn., U.S.A $17 US., all other $19 US.

Canadian Distributor:
Access Computer Services
620B Magnetic Drive
Downsview, Ontario
M3J 2C4
(416) 736 4402

or:

CompuLit
PO Box 352
Port Coquitlam, B.C.
V5C 4K6
604 464 3396

U.S. Distributor
Prairie News
2404 West Hirsch
Chicago, IL
60622
(312) 384 5350

Send all subscriptions to: The Transactor, Subscriptions Department, 500 Steeles Avenue, Milton, Ontario, Canada, L9T 3P7, 416 876 4741. From Toronto call 826 1662

Want to advertise a product or service? Call or write for more information.

Editorial contributions are always welcome and will appear in the issue immediately following receipt. Remuneration is $20 per printed page. Preferred media is 2031, 4040, 8050, or 8250 diskettes with WordPro, WordCraft, Superscript, or SEQ text files. Program listings over 25 lines should be provided on disk or tape. Manuscripts should be typewritten, double spaced, with special characters or formats clearly marked. Photos of authors or equipment, and illustrations will be included with articles depending on quality. Diskettes, tapes and/or photos will be returned on request.

All material accepted becomes the property of The Transactor. All material is copyright by Transactor Publications Inc. Reproduction in any form without permission is in violation of applicable laws. Please re-confirm any permissions granted prior to this notice. Solicited material is accepted on an all rights basis only. Write to the subscriptions address above for a writers package.

The opinions expressed in contributed articles are not necessarily those of The Transactor. Although accuracy is a major objective, The Transactor cannot assume liability for errors in articles or programs.

From The Editor's Desk

Infolution

That's right. Infolution - the evolution of information. Or is that revolution of information. In 1977 (here I go again) when Commodore first released the PET, there was virtually no information available. Then as more questions were asked, more material was generated. Soon there was enough to get by, and after three years plenty for most purposes. But it was scattered and I found myself carrying 20 pounds of stuff around to have maybe 20 pages of it with me. So in 1980, The Transactor released the first Reference Issue, Volume 2, Issue 7.

History repeats itself. When I noticed my briefcase was splitting at the seams, I figured it was time. It would have to be complete, accurate, tidy, and most of all, handy. But in the past 6 years, a lot has been covered, and re-uniting it all into a compact unit would prove to be some task.

Somewhere in the neighbourhood of 2000 hours was spent on this orb. Usually an issue takes about 1 1/2 of the typesetters' 8" disks. This time 6 of them were completely filled. But this one makes the last one typeset by yours truly. Hopefully this will shave some time off our cycle.

You will notice a certain amount of duplicated material as you flip through for the first time. Quite intentional... and the reasons will become more apparent as you use it more. For example, you'll find machine language op-code values presented in at least three spots. Each has its own purpose in life. Butterfield's SuperCharts show the relationship between values as they're interpreted by ASCII, BASIC, the screen, and the microprocessor. I find this chart is most useful when trying to decipher someone elses work (and also anything of my own that's over three months old).

When I'm writing new code, I usually have the 6502 Summary at hand. Alphabetical order makes the instruction I need easier to find, plus the applicable addressing modes and affected status register bits are the kind of information one must have under such circumstances.

The op-code hex values appear again in a rather handy wallet-size chart drafted some years ago by Jim Butterfield. However, if you look between the lines on this one, you'll see a pattern. It was this chart that first introduced me to the structure of the 6502's internal architecture. Notice how 90% of the instructions have been given very deliberate values, not just arbitrary.

Further on the above, the Reference Issue has been assembled with a "same page" philosophy. I admit the characters are a bit small in places, but I find the inconvenience of flipping pages is worth the tradeoff.

Once again I salute Jim Butterfield, the man responsible for all those memory maps and other stuff. The hours of trouble he has saved us all could easily be measured in lifetimes. By the way, we still have copies leftover of Issue 02 featuring a centerfold of the man himself... and in fine form, I might add.

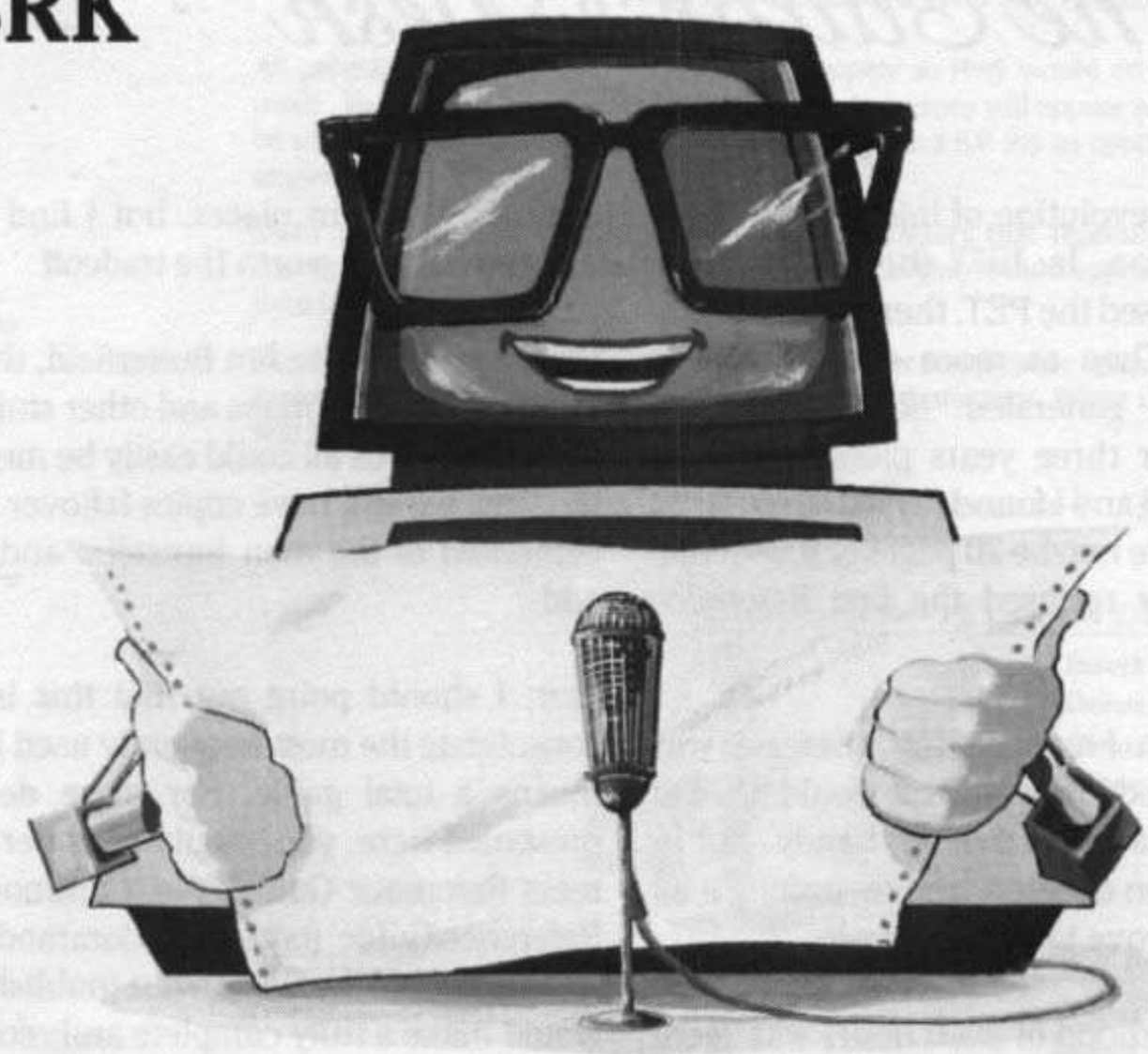
Last, I should point out that this issue was designed to consolidate the most frequently used information. It is by no means a total guide. For more details on the material presented here, you should consider the 'VIC 20 Programmers Reference Guide', the 'Commodore 64 Programmers Reference Guide' (both from Commodore), or 'Programming The PET/CBM' by Raeto West (published by Compute). All 4 would make a truly complete anthology.

And in closing... this issue marks our debut on the newsstand in Canada - we printed 16,000 to meet orders, more than in all of 1979 and '80. And now that typesetting detail is being farmed out, Richard Evers, new full-time Editor, and myself can concentrate on cranking out files instead of film. More writers, articles, circulation, and faster production is now on top of the priorities list. We're paying more now too, call us.

No longer Editor, no longer typesetter, I remain,

Karl J.H. Hildon
Managing Editor, The Transactor

News BRK



The Transactor Adds New Employees And New Features

Richard Evers is now full-time Editor of The Transactor and Karl Hildon moves to the Managing Editors' spot. Both Richard and Karl will continue working on article production. Adding to the traditionally advanced technical content of The Transactor will be more material at the beginner and intermediate levels. This does not mean that advanced level material will be reduced, simply that beginner and intermediate will be increased. This will help broaden the reader base of The Transactor and ultimately lead to increases at ALL levels.

Renewal Time

For many subscribers, this will be the last issue posted against their subscription since The Transactor became independent in November 1982. This notice applies mostly to those that sent less than $15 towards a subscription. If you are one of them, your mailing label will show an asterisk (*) at the top right corner.

If you subscribed starting with Issue 01 and sent the right amount, your last issue will be the next one, Issue 06. However, the postage-free subscription card will NOT be run next issue, so take advantage now. Please indicate 'Renewal' on the card.

No More 04's

Unfortunately, we have run out of Volume 4, Issue 04. Yes, for the first time, we plain ran out! If there is enough demand, we'll print more, but those ordering new subscriptions should consider Issue 05 as their first copy.

Prices in US. Dollars

Lately we've run a couple of ads that show prices with no indication of their currency, ie. Canadian or US. dollars. If the ad has been placed by a firm with a US. address, it should be assumed that the prices are in US. dollars. In future we'll attempt to clarify any ads that don't specify.

COMAL For The 64 Now Public Domain

As of September 30/1983, COMAL for the Commodore 64 has been free public domain software. COMAL was originally written for the PET/CBM. This new rendition has several improvements including commands for controlling colour, sound, and turtle graphics. Best of all, it's free. Contact your club librarian about obtaining copies.

'Home Accountant' and 'Tax Advantage' for C64

Continental Software of Los Angeles, California, in conjunction with their recently appointed marketing firm, South Western Publishing Co., has announced the release of two of their current best selling programs, 'The Home Accountant' and 'The Tax Advantage', for the educational environment. Rewritten specifically for the junior high schools, high schools, universities and adult education classes, the packages all come complete with diskettes, users manuals, teachers manual and test forms. For more information contact:

Hank Scheinberg/Denny Mosier
Continental Software
11223 South Hindry Avenue
Los Angeles, California
90045 (213) 410-3977

Las Vegas Style Gambling On The C64

Advanced Microware has announced the release of their latest effort for the C64, CASINO PAC. It includes four games - Blackjack, Poker, Keno and Slot Machine, and each is said to be accurate simulations of the video game machines currently in use in Las Vega, Reno and Atlantic City. Available on tape or disk for $39.00 US.

Also available is 64TOUR, a demonstration program of the capabilities of the C64. These include High Resolution Graphics, Music and Sound Effects. Priced at $12.00 US. For more information please contact:

Advanced Microware
P.O. Box 6143
Santa Ana, CA.
92786 (714) 554-6470

Save The World From Total Destruction With Your Vic 20

THORN EMI has announced the release of their newest and most challenging game yet, 'Computer War', based on the movie 'War Games'. Though difficult to master, it has been made easier to learn through all stages. Control is accessed either through the keyboard, or with a joystick. For more information please contact:

Nina Budman or Pat McNamara
Budman Math Inc.
505 Eglinton Avenue West, Suite 303
Toronto, Ontario
M5N 1B1 (416) 484-7773

SOLARCON announces 'Passolar', Design Analysis For Passive Solar Systems

Based on the 'Passive Solar Design Handbook', Volume Three from the Los Alamos National Laboratory, Passolar has been hailed as an easy to learn, easy to operate Architects, Consultants and Contractors tool. With input data provided or generated by the client, your desired results are easily attained with the use of this program. Instructions are shown on the screen while the program is in operation, and a very comprehensive users manual is also provided. Available for the Commodore series of business computers, the prices are as follows. $355.00 US with North American Weather Data, and $525.00 US with World Wide Weather Data. For more information please contact:

Norlene H. Martin
SOLARCON, Inc.
607 Church
Ann Arbor, Michigan
48104 (313) 769-6588

Spike-Spikers™, Voltage Suppressors & Noise Filters

Kalglo Electronics Co., Inc. has announced their release of a line of eleven different models of transient voltage suppressors and conducted power line noise filtering devices. Designed to protect, control and organize sensitive high-tech equipment, these units are available at prices ranging from $34.95 to $94.95 US. For further information please contact:

Kalglo Electronics Co., Inc.
Dept. SS
6584 Ruch Road., - E. Allen Twp.
Bethlehem, PA 18017

C.Itoh F10 Printer Cut Sheet Feeder Available For Under $900.00

Canadian General Electric has announced the arrival of its fully mechanical cut-sheet feeder for its C.Itoh F10 Daisywheel Printer. The Model A-100 requires no tools for installation, no power to operate and has no switches or controls at all. Weighing in at ten pounds, this completely metal unit will stack in correct sequence each piece of paper fed to it from its 200 sheet storage bin. Subscripts and superscripts are supported, with a price in total being $850.00 Cdn. For more information:

Mr. Art S. Best
Canadian General Electric - Data Systems
(416) 675-7500

The Grade Manager

'The Grade Manager', for the Vic-20 and the Commodore 64 computers, will alphabetically sort student lists, keep track of assignments, weights, grades, and calculate averages for an entire term. It will report, to TV or printer, assignment summaries, student grades and averages, and incomplete assignments. Available on both cassette and diskette, the disk version is menu driven. For use with the Vic-20, 8K memory expansion is required. Prices are $29.95 US for tape and $34.95 US for disk. For more info, contact

Smokey Mountain Software
54 West Main Street
Brevard, NC
28712 (704) 883-2595

Two New Sports Strategy Games For The VIC 20

Parr Programming, of Gary, Indiana, has produced the first two in a series of Sports Adversary Games for the Vic 20. The first is the Baseball Adversary. With you in the managerial position against your computer, this game allows for additional players on the bench for pinch hitting and relief pitching and is written to not allow your adversary, the computer, any unfair advantage. Control of the game is through the keyboard. Available on cassette and requiring no memory expansion, the price is $10.95 US.

The Football Adversary is the final in this series of two. You are put in the role of head coach, with features being a selection of 31 plays and a display of halftime/game statistics. Available on cassette, and also using keyboard control, the price is $14.95 US.

Parr Programming
2664 Tyler Street
Gary, Indiana
46407 (219) 885-0611

Kobetek Systems Limited announces new STATS software package.

Designed by Mr. Patrick Royston of London, England, Mr. Royston currently holds a masters degree in both statistics and information science. Written for the 4000, 8000/9000 and C64 series of computers, this menu driven program gives the user a completely advanced statistical program for use in any STATS application. With a price of $350.00 to $500.00 Cdn., further interest may be warranted for those who require accurate results in a shorter period of time. For more information contact:

Kobetek Systems Limited
1113 Commercial Street
New Minas, Nova Scotia
B4N 3E6 (902) 678-9800

SWL RTTY/MORSE - Attention Hams and Shortwave Listeners

Let your Commodore 64 or VIC-20 help you 'listen in' on the world of Radioteletype and Morse Code. Converts your VIC-20 and C-64 computer into a low-cost RTTY and morse decoding and display terminal. Allows you to receive 45 to 100 baud BAUDOT RTTY and 5 to 30 wpm morse code. Requires TTL compatible terminal unit such as the MFJ-1224/5 or the HRA Electronics TU-II. Package includes software on cassette, special user port connector, and complete instructions. Price $19.95 US for cassette or $22.95 US for disk plus $2.00 for shipping and handling.

RAK Electronics
P.O. Box 1585
Orange Park, Florida
32067-1585 (904) 264-6777

Modem Protection From Kleen Line

Suppress damaging telephone and power line spikes caused by lightning, spherics or phone office switch gear with the Kleen Line Model PDS-11/SUP. Available for 4 pin telephone modular connectors (RJ-11) and the wider 8 pin connectors (RJ-45), the PDS-11/SUP is priced at $81.95 US. For more information please contact:

Frank Stifter
Electronic Specialists Inc.
171 South Main Street
P.O. Box 389
Natick, Massachusetts
01760 (617) 655-1532

Accounts Receivable Program For The C64

Written by Microspec Inc. of Plano, Texas, this program will support up to 150 customers per diskette on a balance forward system that will accommodate 18 transactions per customer for each posting period. Complete aged reports are generated and automatically posted to Microspecs General Ledger program. The price for this program is $99.95 US. with more information available from:

Laura Robinson
Communications Plus
301 S. Sherman
Suite 117
Richardson, Texas
75081 (214) 783-8543

Allow Your Vic or C64 To Balance Your Chequebook

Microspec of Plano Texas, in its never ending quest for further C64 and Vic 20 programs, has released the 'Checkbook Manager'. Available for both the C64 and Vic 20, each appear to offer a solution to the problem of keeping your chequebook in order. Able to track all outstanding cheques, reconsile monthly financial statement, and track expenses on a quarterly and year-to-data basis, the limit only appears to be the capacity of the version purchased. The Vic 20 version offers ten less categories than that of the C64. The price is $69.95 US for the C64 version, and $49.95 US for the Vic 20. For further information please contact Communications Plus (above).

'Power 64' Available For The Commodore 64

In the past we have all come to know and love the features of Power for our Commodore computers. But with the advent of the Commodore 64, a loss has been felt. After being spoiled by Power, BASIC programming has suddenly become a chore. No longer must this be, for Brad Templeton has re-written his utility for the C64.

In case you have never heard of Power, the following are some of the features it has to offer.

- automatic line numbering and re-numbering
- complete tracing functions
- single stepping through programs
- definition of keys as BASIC keywords
- de-bugging ease with a 'why' command
- addition of auto repeat function to cursor keys
- ability to merge BASIC programs
- listing and scrolling up and down of BASIC programs in memory
- hexadecimal and decimal conversions
- and much, much more

The documentation included, written by Jim Butterfield, is very helpfull in all aspects and leaves nothing to the imagination. For a total price of $99.95 Cdn., the benefits shown will be worth the cost involved. For more contact:

Pro-Line Software Ltd.
755 The Queensway East, Unit 8
Mississauga, Ontario
L4Y 4C5 (416) 273-6350

The MANAGER Column

Don Bell & Alex Gaul
Milton, Ontario

In this issue, the Manager column is devoted to the first part of a hands-on application for the '64 MANAGER - a Christmas card and gift list. The second part of this article will appear in the next issue. Subsequent issues will provide more easy to follow hands-on applications for home and business as well as tips for advanced users including "Re-Field' by Eric Armson, a routine you add to Create/Revise that re-orders field entry. - Ed.

A '64 MANAGER Gift List

HUMBUG! It's that time of year again. I staggered up the stairs from the basement dragging the box of old Christmas junk accumulated over years of neglect. Then after a stiff drink, I screwed up enough courage to open it. YUK! Krinkled tinsel, tangled Christmas tree lights, broken decorations, horrible-looking holly, musty mistletoe (smelling of old kisses), petrified candy canes and at the bottom, boxes of last year's Christmas cards. Cheer up, the worst is yet to come!

Then, like a memory of a NEW Year's Day hangover, it suddenly hit me! How am I going to cope with the Christmas cards, gifts, and party again this year? There's a number of people I'd like to 'edit' from my list but I can never remember whether they sent me a card or a gift last year. (I'd prefer not to expose myself as an incompetent who can't even master a Christmas card list.) Usually what happens is I send them a card one year, they send me one the next... and so on. It's as though each of us wants to send the last missile... I mean card. Then what to do about gifts - especially those useless token gifts to appease distant relatives. I've got enough socks & ties to dress a herd of caribou and enough after-shave lotion to bathe in. I'd trade it all for one good Darth Vader video game. It gets down to a matter of how much to spend - an equation - whatever they spend I have to spend. Boring! Then who to invite to the party? How to avoid inviting people who mix like sharks and jellyfish. How will I get through it all again without losing my mind?.

HARK! This year I decided there had to be a better way. I want to get rid of the worry, the planning, the strain on my old memory bank and get back some real Christmas cheer, brouhaha, whatever it was that made old Scrooge want to click his heels on Christmas Day. Taking my cue from a business associate of mine who is ecstatic over his new computer system, I decided to try my hand at solving my problem using THE '64 MANAGER.

Very slowly and carefully I will show you the steps I took creating my Christmas List file. You will be able to easily keep track of cards, gifts, parties, money spent, and other Christmas information. You will be able to generate numerous useful reports that would be difficult or impossible to create without it. All of this is directed towards helping you better organize yourself, so that you will have more time to experience the real joys of Christmas.

LOADing The Program

Turn on your 64, monitor and disk drive.
Insert your Manager disk in the drive
Type: load "64.manager", 8 press 'return'.
When the cursor reappears, type in 'run' and press 'return'.
After a few minutes, a menu comes up on the screen and you are ready to start.

Formatting A Diskette

The first thing you must do is "format" a blank diskette, ie. prepare it so it can receive data. Press 'f' to format a disk. The computer will load the format program from the disk. When the computer has finished loading, remove THE MANAGER diskette and insert your blank data diskette into the drive. Follow the screen instructions and you will have a formated disk.

Getting Started In The CREATE/REVISE Option

When the main menu appears, press 'c' for create/revise. This option lets you draw a form on the screen and revise it later if you choose. The first screen prompt is 'filename?'. I called my file 'xmas list'. Enter any filename you want (preferably one that reminds you of the file) and press 'return'. The next screen prompt is 'create using an existing screen (y/n)?'. Enter 'n' as you do not have an existing screen.

Selecting Screen Colours

Since this is a Christmas List and not an accounting application, you can choose any wild, cheerful colours you wish for your screen.

The system will now prompt:

ENTER BORDER COLOUR?
ENTER BACKGROUND COLOUR?
ENTER CURSOR COLOUR?

For each prompt select a colourr by pressing 'CTRL' and a number key with a colour name on its side, OR, 'Commodore KEY' and a number key with a colour name on its side

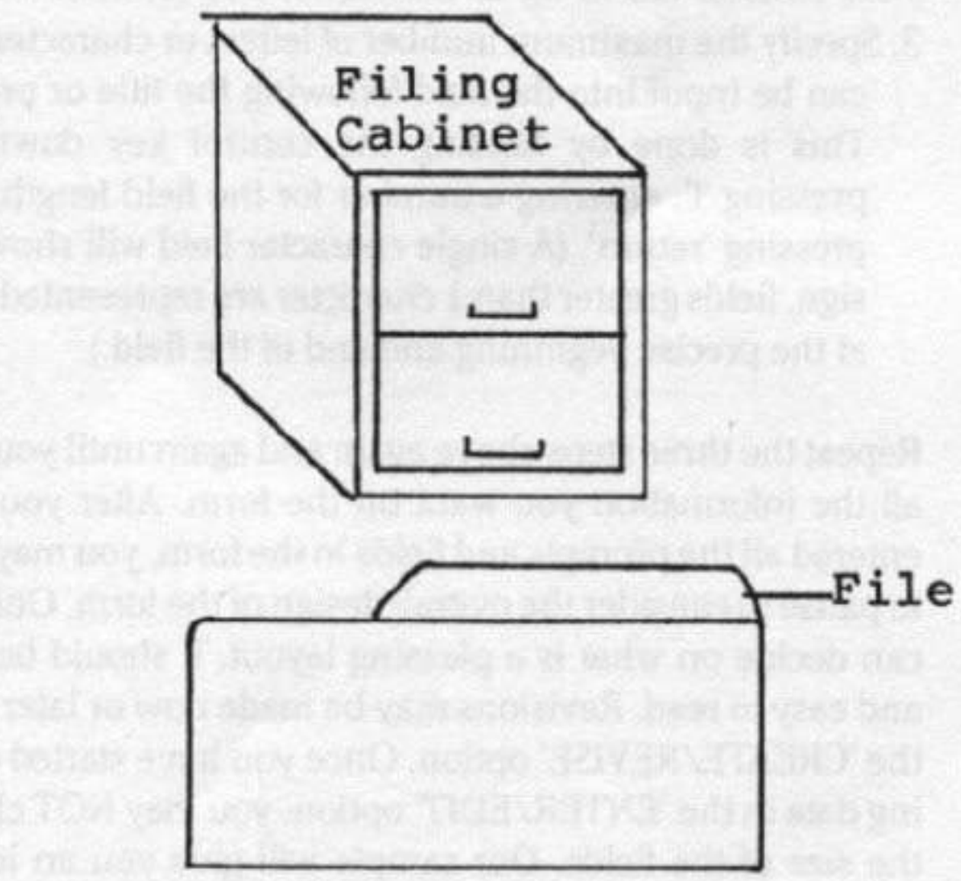
Note that if you chose a black background with black border and black cursor nothing would appear on the screen. This can be remedied by choosing a light cursor colour ie. white or yellow. Refer to the instruction manual that came with your 64 for further details.

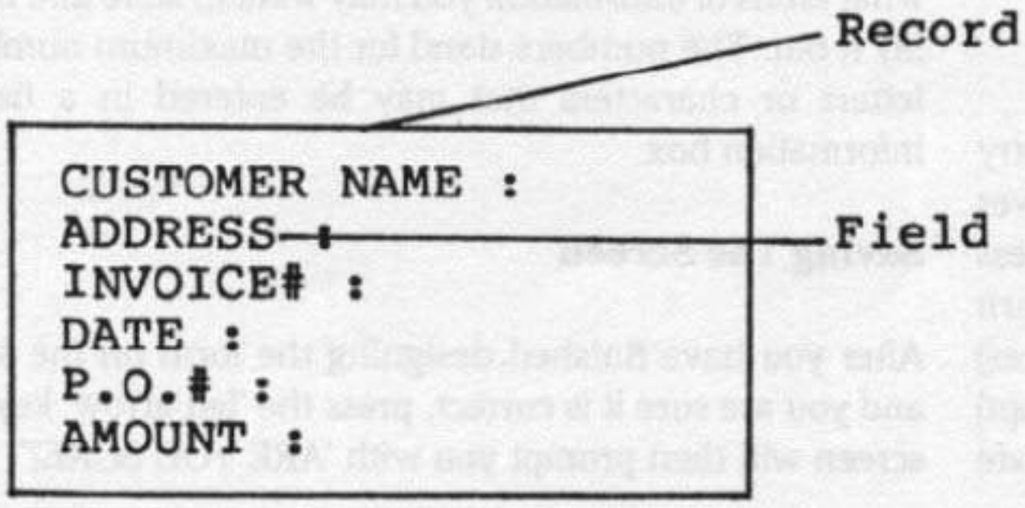
Designing A Form For Your Christmas List Data

After choosing your screen colours and pressing 'return', your screen will be cleared except for the bottom line where it will say screen '1 of 1' and 'L1 C1'. This means that you are on screen 1 and your cursor is on line 1 column 1. You can now create a screen for your Christmas list.

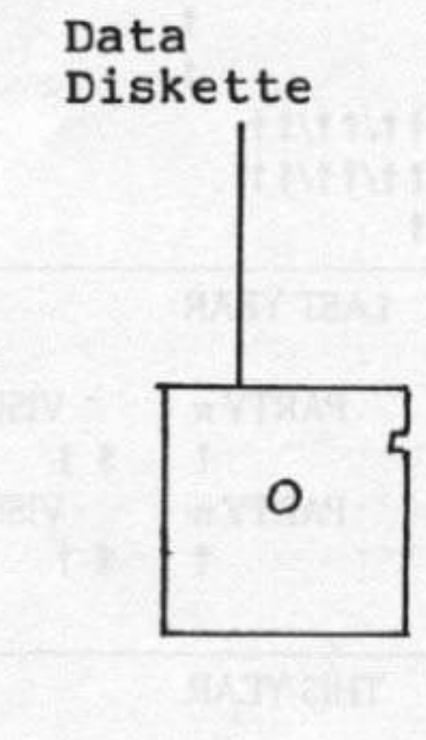
Picture a file folder filled with blank pieces of paper. The file folder is really our empty data diskette and the blank pieces of paper are blank screens with nothing on them. You are now going to create 'forms' out of the blank pieces of paper.

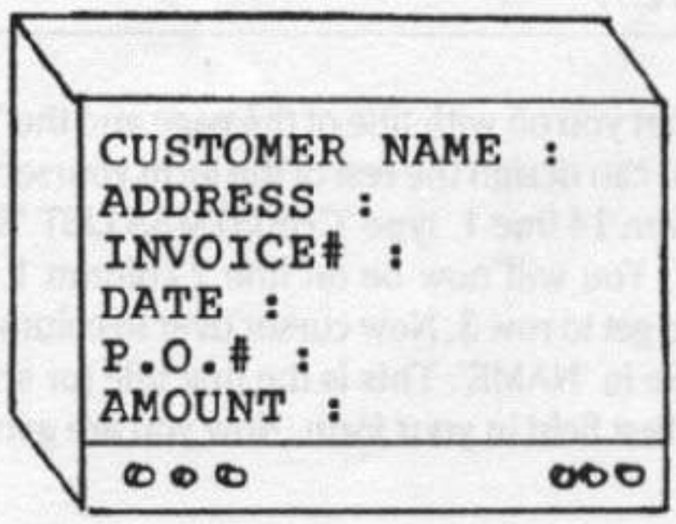
MANUAL FILING SYSTEM





'64 MANAGER FILING SYSTEM





Picture a 'form' with titles and boxes to fill in. The top of a job application, similar to the one below, may help you visualize a form.

JOB APPLICATION FORM

NAME : _____

ADDRESS : _____

Notice that a form has titles or prompts followed by specific areas in which to enter information. We are going to use a form for entering information in our Christmas List.

Using the screen as a blank piece of paper, you will now design a form that accomodates all the information you need to know about a person on your Christmas List. When you are finished it may resemble the screen below. Each '↑' indicates the beginning and end of a box or field of information. The 'π' represents a field that will only need 1 character of information.

CHRISTMAS LIST '83

```
NAME          ↑                                  ↑  M/F π
ADDRESS       ↑                                     π
CITY          ↑                                     ↑
STATE         ↑                                     ↑
ZIP           ↑                    ↑
PHONE         ↑                    ↑
BIRTHDAY            ↑↑/↑↑/↑↑
LAST CONTACT        ↑↑/↑↑/↑↑
NOTES               ↑                              ↑
```

LAST YEAR

```
REC'D: CARD π        PARTY π        VISIT/CALL π
       GIFT  ↑              ↑     $ ↑            ↑
GAVE:  CARD π        PARTY π        VISIT/CALL π
       GIFT  ↑              ↑     $ ↑            ↑
NOTES↑                                          ↑
```

THIS YEAR

```
CARD π          PARTY INVIT. π        VISIT/CALL π
       GIFT  ↑              ↑     $ ↑            ↑
NOTES↑                                          ↑
```

I will start you off with title of the page and the 'NAME' entry and you can design the rest of the form yourself. Cursor over to column 14 line 1, type 'CHRISTMAS LIST '83, then press 'return'. You will now be on line 2 column 1. Press return again to get to row 3. Now cursor over to column 4 (3 spaces) and type in 'NAME'. This is the first title (or screen prompt) for the first field in your form. Now you are going to indicate the field size. While holding the CONTROL key down, press 'f'. After releasing the keys the prompt 'LENGTH?' will appear on the bottom line. The program will accept numbers between 1 and 30 for the length of the field. In our case, we have chosen 20 characters as the length of the 'name' field. Enter '20' and press 'return'. You will notice that this command will leave two ↑'s to indicate where the field is. The length of the field on the screen includes the two ↑'s. (The ↑'s may also be keyed in directly, but you will have to count the field length yourself. Be sure to include the ↑'s in your field count.)

In a similar manner to the above, fill in the rest of the form repeating the 3 steps below for each piece of information.

1. Decide on a specific piece of information you want included on the form. (Information such as name, address, phone, whether you want to send them a card or gift, etc.)

2. At a suitable spot on the screen, enter a title or prompt for that piece of information to be input, eg. "NAME" prompts the user to input a name. There must be sufficient room on the right side of the prompt to enter the required information.

3. Specify the maximum number of letters or characters that can be input into the field following the title or prompt. This is done by holding the control key down and pressing 'f', entering a number for the field length, then pressing 'return'. (A single character field will show a 'π' sign, fields greater than 1 character are represented by ↑'s at the precise beginning and end of the field.)

Repeat the three steps above again and again until you have all the information you want on the form. After you have entered all the prompts and fields in the form, you may want to pause to consider the overall design of the form. Only you can decide on what is a pleasing layout. It should be clear and easy to read. Revisions may be made now or later using the 'CREATE/REVISE' option. Once you have started entering data in the 'ENTER/EDIT' option, you may NOT change the size of the fields. Our sample will give you an idea of what kinds of information you may want to store and how to lay it out. The numbers stand for the maximum number of letters or characters that may be entered in a field or information box.

Saving The Screen

After you have finished designing the form on the screen and you are sure it is correct, press the 'left arrow' key. The screen will then prompt you with 'ARE YOU SURE?'. Enter

'y' and press 'return'. The program will check to see if all fields are closed (fields longer than 1 character must have a pair of ↑'s. If there is an error, you will be prompted to change the field entry.

Next the screen will prompt 'Another Screen?'. In our case we will enter 'n' and press 'return'. Then the screen will prompt 'Do you wish to alter field types?' this is to change between an alpha field ie. letters of the alphabet and numeric fields ie. numbers. As we do not really have any purely numeric fields, we will leave all the fields the same by entering 'n' and 'return'.

Deciding On The Number Of Records For Your File

Now place the data disk you previously formatted into the disk drive after removing the manager disk. The screen will prompt you with 'Rec Len XXXX, Max No of Rec XXXX' at the bottom of the screen and asks 'OK?'. Enter 'y' and 'return'. Then the screen will prompt 'Rec Len,# of records' at which point you can accept the number of records displayed by pressing 'return' or enter your own number of records (up to about 300) and press 'return'. The computer will then go to the disk and create a file large enough to handle all the records you asked for. As this process will take some time, you may want to go to the kitchen for a snack.

Entering Individual Records In Your File

Now you may enter the individual records for people on your Christmas List using the form you have just designed on the screen. Choose 'Enter/Edit' on the main menu. In this option you may enter, change, delete and search for records of individual people in your Christmas List. (More advanced commands will be discussed in future articles.) For each person on your Christmas List, complete a screen form with the following steps:

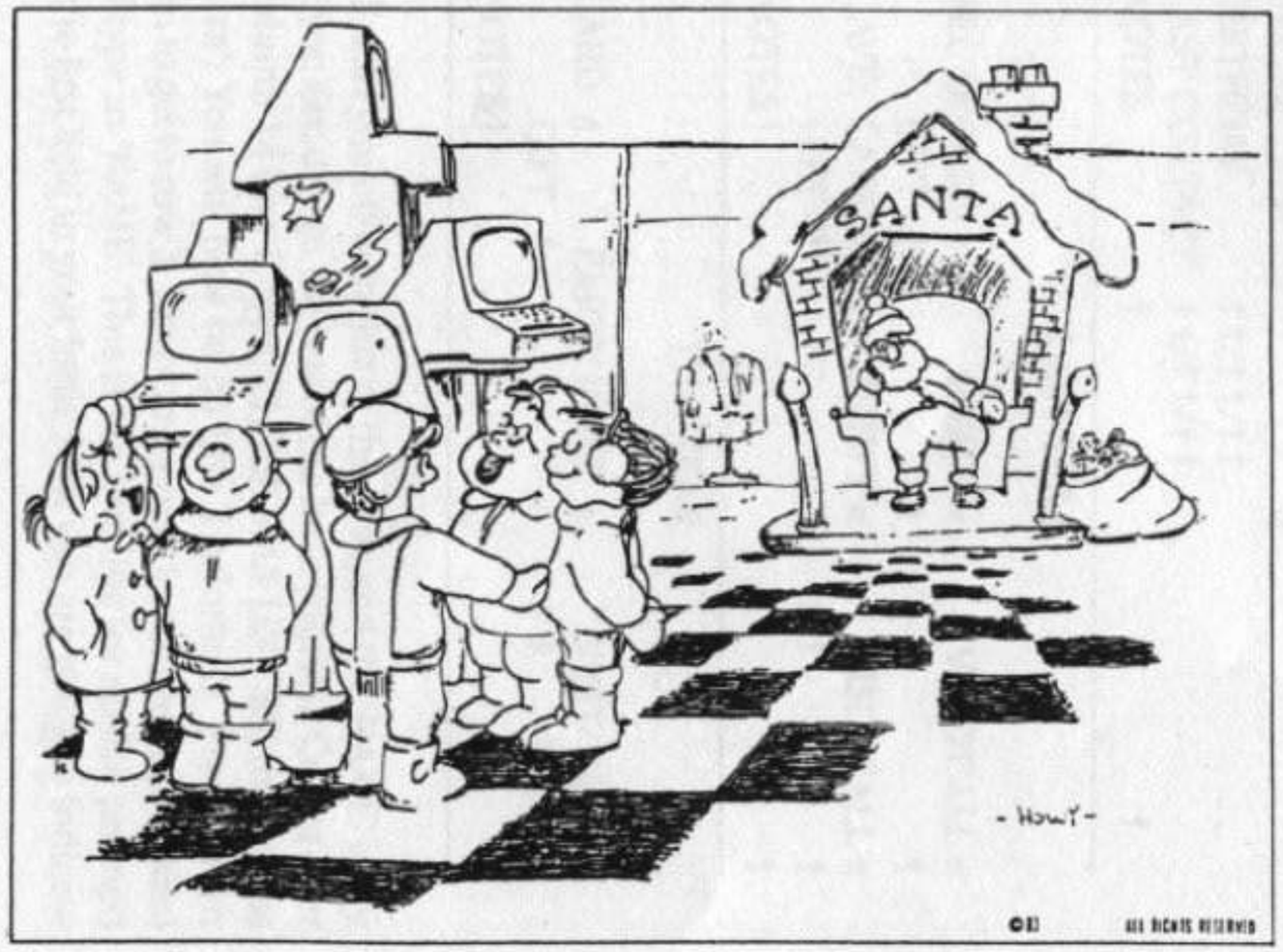
1. Press 'e' to enter a record;
2. Fill in the blank fields, pressing 'return' after each entry;
3. Press ← to save the record. When you have entered a record for every person on your Christmas list, press F1 to return to the main menu. The screen will prompt 'Are you sure (y/n)?', enter 'y', 'return'.

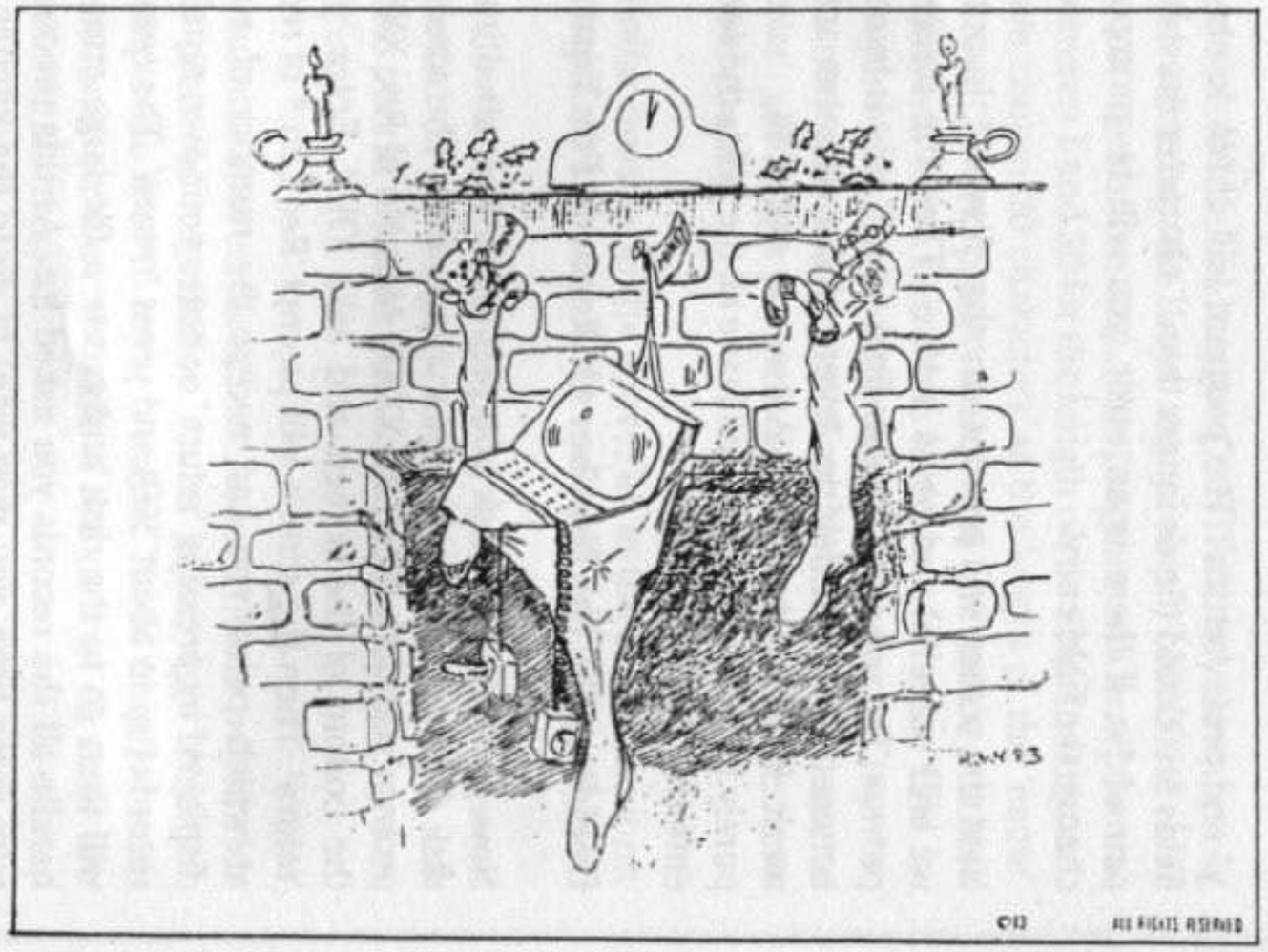
More To Come

In the next article, we will discuss how to do sorts, index searches and generate many useful lists such as a Christmas card list, a gift list, a party list, and a birthday reminder list.

Merry Christmas

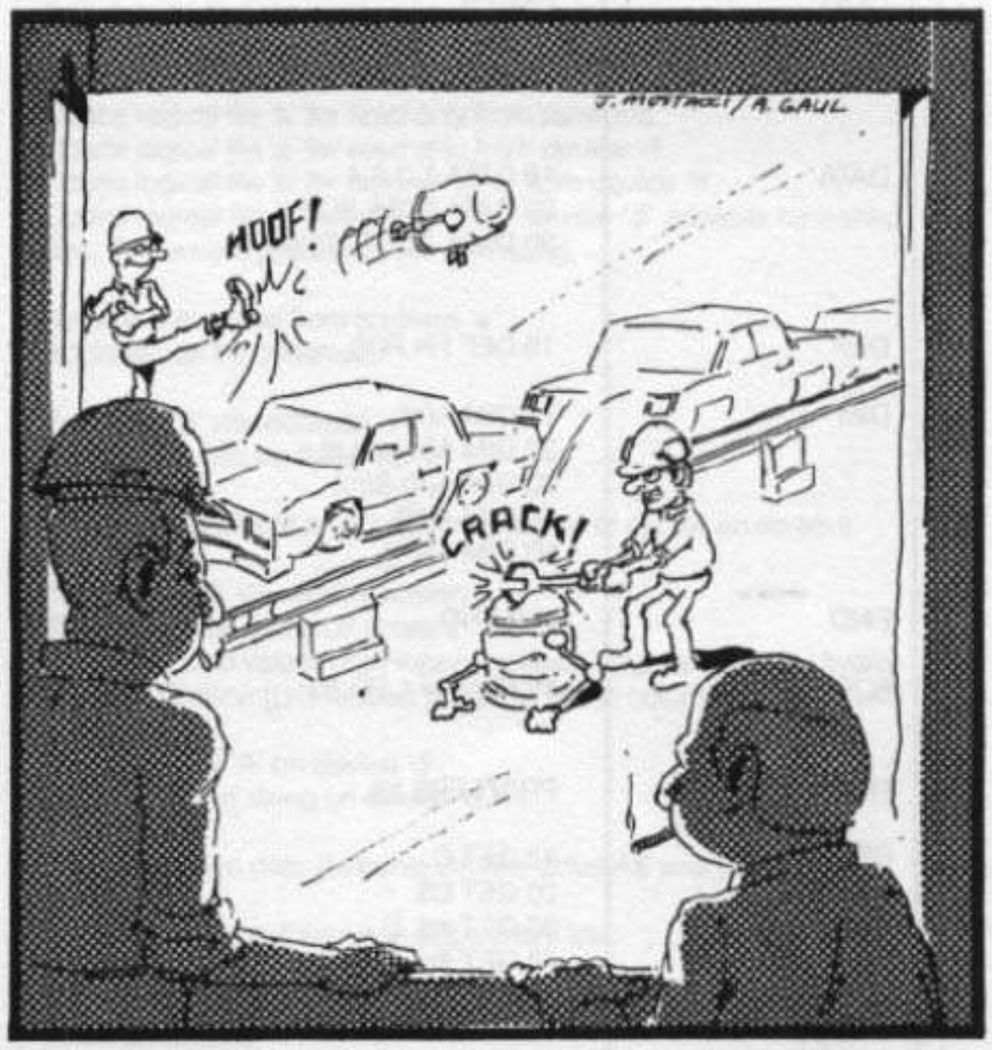
from Mike Panning & Howy Parkins



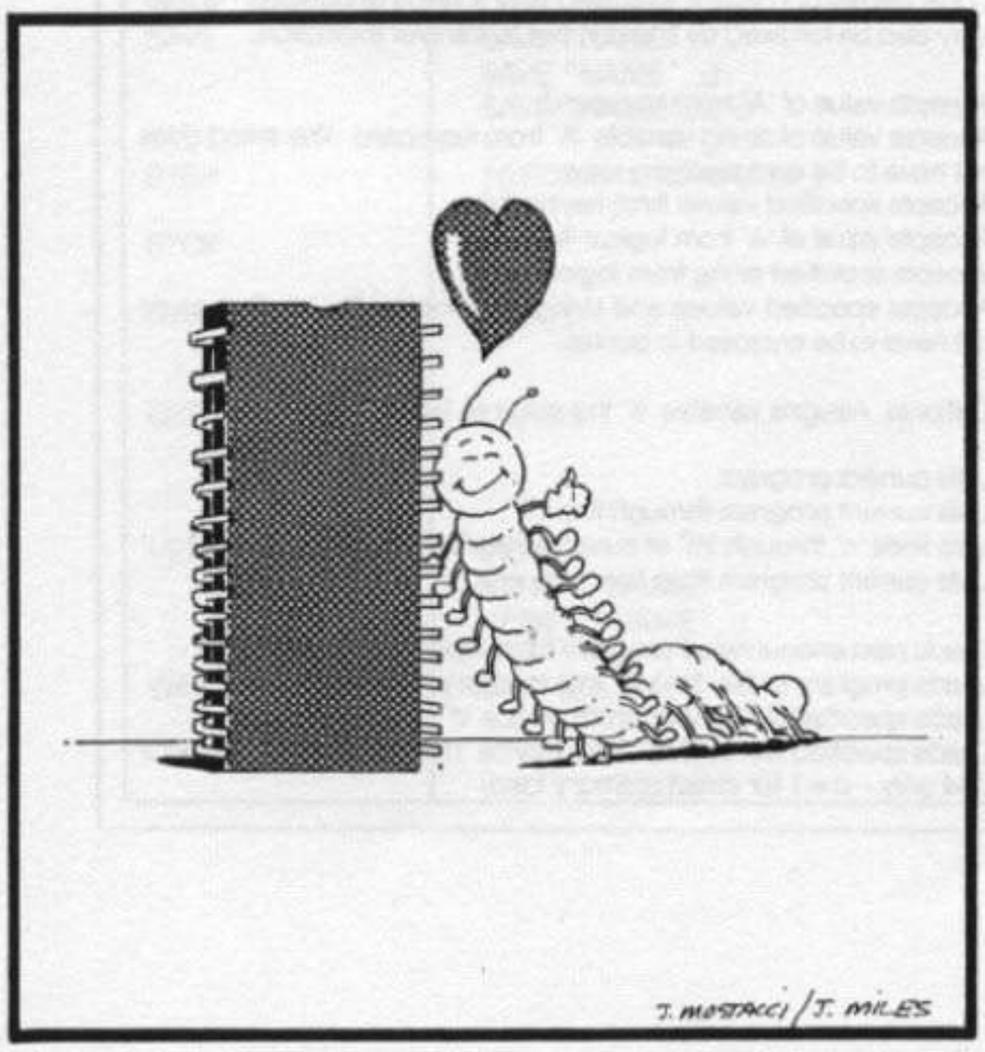


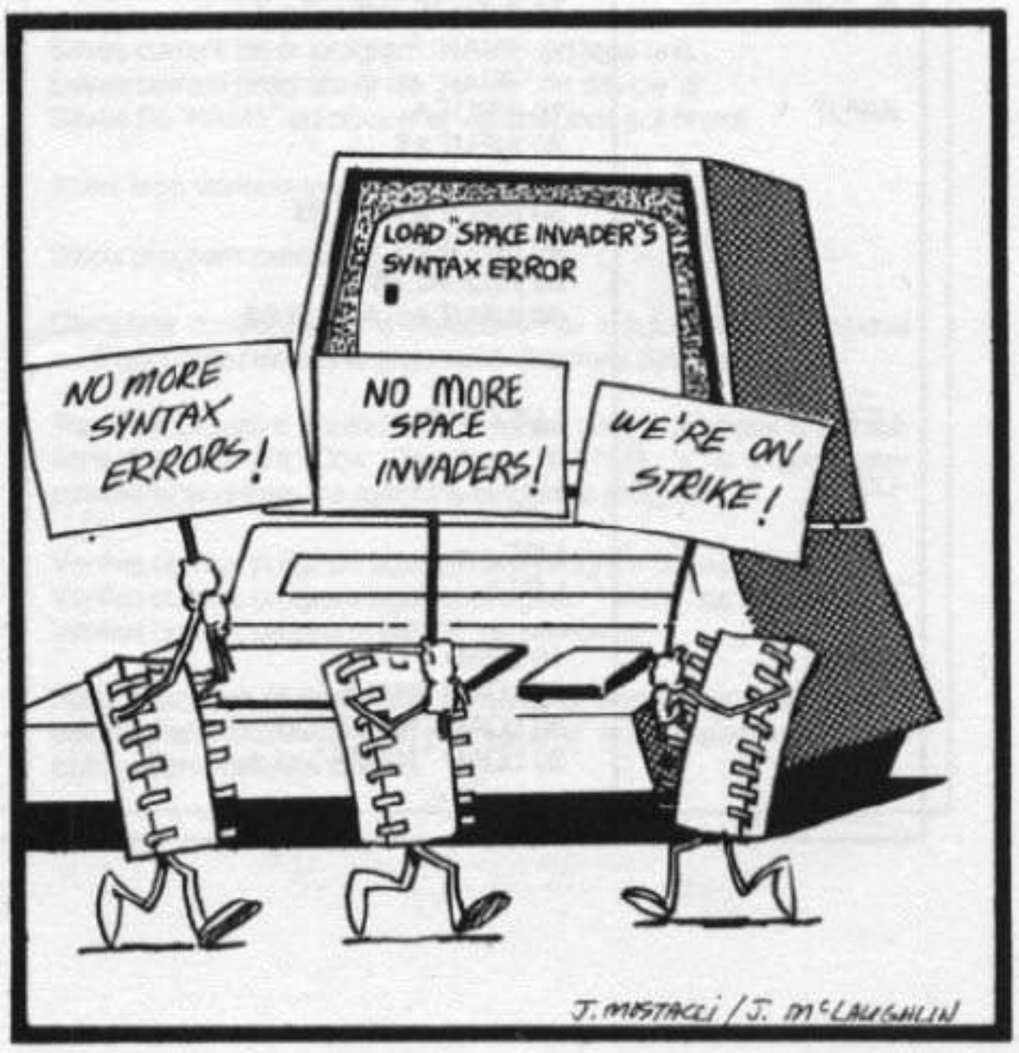
CompuKinks.

Don't worry Mrs. Smith, his facial muscles will eventually relax.
But really, he shouldn't be playing video games
for such long periods of time.



I see the Union is
laying off more robots





BASIC – Beginners All–Purpose Symbolic Instruction Code

Commands, Statements, and Symbols

Command/ Statement	Example	Purpose
CLOSE	10 CLOSE n	Closes logical file 'n'.
CLR	CLR	Sets variables to zero or null.
CMD	CMD D	Keep ieee device 'D' open to monitor bus.
CONT	CONT	Continue program execution after a stop command. No program changes are permitted.
DATA	10 DATA 1,2,3,4 20 DATA TOM, SUE 30 DATA "DOE, TOM"	Specifies data to be read left to right. Alphabetics do not need to be enclosed in quotes. if strings contain spaces, commas, colons, or graphic characters, the string must be enclosed in quotes.
DEF	10 DEF FN R(X)	Defines function 'R'
DIM	10 DIM A(n) 20 DIM A(n,m,o,p) 30 DIM A(n),B(m) 40 DIM A(N) 50 DIM A$(n)	Specifies maximum number of elements in an array or matrix. Specifies maximum number of dimensions in an array. Number of arrays limited by memory. May be dimensioned dynamically. Strings to be dimensioned.
END	999 END	Terminates program execution.
FOR	10 FOR I = 1 TO 10	Begins repetitve loop, specifying loop variable and number of intended iterations (in this example 'I' for 10 iterations).
FRE	PRINT FRE (0)	Returns number of bytes of available memory.
GET	10 GET C 20 GET C$ 30 GET #d, C 40 GET #d, C$	Accepts single character from keyboard. Accepts single string character from keyboard. Accepts single character from specified logical file. Accepts specified single string character from logical file.
GOSUB	10 GOSUB n	Begins execution of a subroutine which begins on line 'n'.
GOTO	GOTO n	Continue program execution at line n after a stop command. Program changes are permitted.
IF...GOTO	10 IF X = 10 GOTO n	Transfers execution to line 'n' if result of condition is true.
IF...THEN	10 IF X = 10 THEN Y = 3	Code following THEN is executed only if result of condition is true. May also be followed by line number to transfer execution.
INPUT	10 INPUT A 20 INPUT A$ 30 INPUT A,A$,B,B$ 40 INPUT #d, A 50 input #d, a$ 60 INPUT #d, A,A$,B,B$	Accepts value of 'A' from keyboard. Accepts value of string variable 'A' from keyboard. The string does not have to be enclosed in quotes. Accepts specified values from keyboard. Accepts value of 'A' from logical file 'd'. accepts specified string from logical file 'd'. Accepts specified values and string from logical file 'd'. Strings do not have to be enclosed in quotes.
LET	LET X = 10	Optional. Assigns variable 'X' the value of 10.
LIST	LIST LIST -n LIST n-m LIST n-	Lists current program. Lists current program through line 'n'. Lists lines 'n' through 'm' of current program. Lists current program from line 'n' to end.
LOAD	10 LOAD 20 LOAD "NAME" 30 LOAD "NAME", d 30 LOAD "NAME", d, c	Loads next encountered program from tape unit into memory. Loads program or file 'NAME' into memory from tape unit. Loads specified file 'NAME' from device 'd'. Loads specified file 'NAME' from device 'd' for command 'c'. (VIC/C64 only – c = 1 for direct memory load)

Command/ Statement	Example	Purpose
NEW	NEW	Deletes current program in memory, sets variables to zero.
NEXT	NEXT	Indicates end of code contained in a FOR/NEXT loop.
ON...GOSUB	10 ON A GOSUB l, m, n	Begins execution of subroutine which begins on specified line (in this example, 'l','m', or 'n') depending on value of index 'A'.
ON...GOTO	10 ON A GOTO l, m, n	Transfers control to specified line 'l','m', or 'n' depending on value of index 'A'.
OPEN	10 OPEN a 20 OPEN a, d 30 OPEN a, d, c 40 OPEN a, d, c, "NAME"	Opens logical file 'a' for read only from tape unit. Opens logical file 'a' for read only from device 'd'. Opens logical file 'a' for command 'c' from device 'd'. Opens logical file 'a' on device 'd'. If device 'd' accepts formatted files, file name is positioned for command.
PEEK	PEEK(a) PEEK(A)	Returns byte value from address 'a'. Address can be dynamic.
POKE	POKE a, b POKE A, B	Puts byte 'b' into address 'a'. Parameters can be dynamic.
POS	10 PRINT POS(0)	Prints next available print position (position of cursor on screen).
PRINT	10 PRINT A 20 PRINT A$ 30 PRINT A, A$ 40 PRINT #d, A 50 PRINT #d, A$	Prints value 'A' on display screen. Prints specified string on screen. Prints specified values or strings on screen, beginning in next available print position (pre-tabbed positions are in columns 10,20,30,40 etc.). Prints value of 'A' on device 'd'. Prints specified string on device 'd'.
READ	10 READ A$, B$	Reads next two data elements into variables A$ and B$.
REM	10 REM comment	Remark indicator. Execution skips entire line.
RESTORE	10 RESTORE	Resets data pointer so that next READ receives first element of first DATA statement.
RETURN	9990 RETURN	Subroutine exit; transfers control to the statement following most recent gosub directing transfer to the subroutine.
RUN	RUN RUN n	Begins execution of program at lowest line number. Begins execution of program a line 'n'.
SAVE	SAVE "NAME" SAVE "NAME", d SAVE "NAME", d, c	Saves current file or program 'NAME' on tape unit. Saves current program or file 'NAME' on device 'd'. Saves file 'NAME' on device 'd'. 'c' specifies eof or eot.
STEP	10 FOR I = 1 TO 10 STEP 2	Alters loop variable increment.
STOP	STOP	Stops program execution.
SYS	SYS (x)	Complete control of pet is transferred to a subsystem at decimal address contained in the argument. Brackets optional.
USR	USR (x)	Transfers program control to a program whose address is at locations 1 and 2 (VIC/C64 - locations 784,785). 'x' is a parameter passed to and from the machine language program.
VERIFY	VERIFY VERIFY "NAME" VERIFY "NAME", d	Verifies current program against next program on tape unit. Verifies current program against program 'NAME' on tape unit. Verifies current program 'NAME' on device 'd'.
WAIT	WAIT a, b, c	Halts execution of Basic until contents of address 'a', and'ed with value 'b' and exclusive or'ed with value 'c', is not equal to zero. 'c' is optional and defaults to zero.

Arithmetic Operators

Symbol	Example	Purpose
=	10 A = B	Assigns a value to a variable.
↑	30 PRINT A↑2	Exponentiation
/	40 C = A/8	Division.
*	50 C = A*8	Multiplication.
+	60 C = A + 8	Addition.
-	70 C = A–8	Subtraction.
=	10 IF A = B THEN PRINT C	'A' Equals 'B'.
<>	10 IF A<>B THEN C = 4	'A' Does not equal 'b'.
<	10 IF A<B THEN C$ = "X"	'A' Is less than 'B'.
>	10 IF A>B THEN C$ = "Y"	'A' Is greater than 'B'.
<=	10 IF A<=B THEN C = 20	'A' Is less than or equal to 'B'.
>=	10 IF A>=B THEN C = D-1	'A' Is greater than or equal to 'B'.
AND	10 IF A AND B THEN C = 9	'A' and 'B' must both be true for statement 10 to be true.
OR	20 IF A OR B THEN C = 9	'A' must be true or 'B' must be true for statement 20 to be true.
NOT	30 IF NOT A THEN PRINT C	Expression is true if 'A' is false.

Note: the numerical values used in the evaluation of logical comparisons are: 'true' is any non-zero number and 'false' is zero.

Special Symbols

Symbols	Example	Purpose
:	10 A = 1:B = 2:C = 3	Allows multiple statements on a line.
;	10 PRINT A;B 20 PRINT A$;B$	Allows same line printing. Elements are separated by 3 spaces. Allows same line printing. String elements are concatenated.
.	X = 10.99	Decimal point
,	10 PRINT A, B LOAD "NAME",d	Allows same line printing. Elements are separated and printed in pre-'tab'ed print positions (columns 10,20,30, etc.). Separates parameters in load, save, open, mid$, on..goto, etc.
?	10 ?A	Abbreviation for 'print'. Stores as one character; lists as word PRINT.
$	10 A$ = "ABCDEFG"	String identifier.
%	10 A% = INT(X)	Integer identifier.
#	10 PRINT#8	Logical file number indicator
"	10 A$ = "ABCDEFG"	String enclosures.
()	X = (A–2)/(B + 2)	Expression priority evaluation
π	10 C = π*D	Value of Pi 3.1415927.

Reserved Variables

Variable	Purpose
DS	Disk Status number (except 2.0)
DS$	Disk Status string (except 2.0)
EL	Error Line (B Series only)
ER	Error number (B only)
ERR$(	Error String array. See table for messages. (B only)
TI	Time in Jiffies (1/60th's sec.) since power up or TI$ reset (except B Series)
TI$	Time in HHMMSS
ST	The Status variable. See table for functions.

Hierarchy of Operations

Operator	Description
( )	Brackets always dictate priority
↑	Exponentiation
-	Negation (unary minus)
* /	Multiplication & Division
+ -	Addition & Subtraction
< = >	Relational Operations
NOT	Logical NOT (Integer two's complement)
AND	Logical AND
OR	Logical OR

Basic 4.0 Disk Commands

Function	Example	Purpose
APPEND	10 APPEND#d, "NAME"	Open file 'NAME' on device 'd' for appending. New data is added to end of existing data.
BACKUP	BACKUP D0 TO D1	Duplicate disk in drive 0 onto disk in drive 1
CATALOG	CATALOG D0	Displays list of filenames in specified drive.
COLLECT	COLLECT D1	Purges disk in specified drive of any improperly closed files (indicated by * beside file type).
CONCAT	CONCAT "NAME1" TO "NAME2", D1	Concatenates file "NAME1" to file "NAME2". Ie. NAME2 = NAME2 + NAME1
COPY	COPY "NAME",D0 TO "NAME",D1 COPY "NAME",D0 TO "DUP",D0 COPY D0 TO D1	Copies file "NAME" from drive 0 to drive 1 Makes duplicate of file "NAME" Copies entire contents from D0 to D1
DCLOSE	DCLOSE #n	Closes disk logical file 'n'
DIRECTORY	DIRECTORY D0	Exact same as Catalog. Use preference.
DLOAD	DLOAD "NAME",Dd,Uu	Loads program "NAME" from drive 'd' on unit 'u'
DOPEN	DOPEN#n, "NAME",Dd,Uu DOPEN#n, "NAME",Dd,Uu,W	Opens file "NAME" for reading from drive 'd', unit 'u'. Default values: d=0, u=8. Data is retrieved through file number 'n'. Opens file "NAME" for writing to drive 'd', unit 'u'. Not necessary for RELative files.
DSAVE	DSAVE "NAME",Dd,Uu	Saves current program to drive 'd' on unit 'u' as file "NAME"
HEADER	HEADER"DISKNAME",Dd,Iid,Uu	Formats disk in drive 'd' unit 'u' assigning it a "DISKNAME" and 'id'.
RECORD	10 RECORD#n, a	Positions relative file open on logical file number 'n' to record number 'a'. 'a' may be dynamic but must be enclosed in brackets.
RENAME	RENAME "NAME" TO "NEWNAME",D0	Changes a file name.
SCRATCH	SCRATCH "NAME",D1	Eliminates file "NAME" from disk.

Additional B Series Commands

Function	Example	Purpose
BANK	BANK b	Sets bank number to 'b'.
BLOAD	BLOAD "NAME",Dd,Uu,ON Bb,Pp	Loads file "NAME" from drive 'd' unit 'u' into bank 'b' at position 'p'
BSAVE	BSAVE "NAME" ON Bb,Pp1 to Pp2	Saves current memory in bank 'b' from address 'p1' to 'p2' as file "NAME" to drive 0 unit 8. Addresses are in decimal.
DCLEAR	DCLEAR D1	Initialize disk in drive 1
DELETE	DELETE 10-30	Deletes lines from current program. Specify line range same as LIST.
DISPOSE	DISPOSE GOSUB	Purges stack of unwanted return addresses (like 'POP')
ELSE	IF ST THEN E=1 ELSE E=0	Alternate condition following IF..THEN. May also be used to transfer execution
INSTR	PRINT INSTR (A$, B$)	Returns position of string B$ within A$. Returns 0 if not found.
KEY	KEY KEYn, "CATALOG D0" + CHR$(13)	Displays list of function key definitions Defines function key 'n'.
PUDEF	PUDEF " -.£"	Re-defines Print Using format characters. Default is " ,.$". In this example, space is changed to '-', comma to period, period to comma, and dollars to pounds.
RESUME	RESUME RESUME n RESUME NEXT	Continues execution after program error or editing Resumes execution at line 'n' Resumes execution at start of current active FOR/NEXT
TRAP	TRAP 50000	Specifies routine at line 50000 as an ON ERROR routine.
USING	PRINT USING "-$##,###";X	Specifies format to be used for numerical output.

String Functions

Function	Example	Purpose
ASC	10 A = ASC(" XYZ ")	Returns the integer value corresponding to ASCII code of the first character in string.
CHR$	10 A$ = CHR$(n)	Returns character corresponding to ASCII code number.
LEFT$	10 PRINT LEFT$(X$, a)	Returns leftmost 'a' characters from string.
LEN	10 PRINT LEN(X$)	Returns length of string.
MID$	10 PRINT MID$(X$, a, b)	Returns 'b' characters from string, starting with the 'a'th character.
RIGHT$	10 PRINT RIGHT$(X$, a)	Returns rightmost 'a' characters from string.
STR$	10 A$ = STR$(A)	Returns string representation of variable 'A'
VAL	10 A = VAL(A$) 20 A = VAL(" A ")	Returns numeric representation of string. If string not numeric, returns " 0 ".

ASC, LEN and VAL functions return numeric results. They must be used as part of any expression. Assignement statements are used here for examples only; other statement types may be used.

Arithmetic Functions

Function	Example	Purpose
ABS	10 C = ABS(A)	Returns magnitude of argument without regard to sign.
ATN	10 C = ATN(A)	Returns arctangent of argument. 'c' will be expressed in radians.
COS	10 C = COS(A)	Returns cosine of argument. 'A' must be expressed in radians.
DEF FN	10 DEF FNA(B) = C * D	Allows user to define a function. Function label 'a' must be a single letter; argument 'b' is a dummy.
EXP	10 C = EXP(A)	Returns constant 'e' raised to the power of the argument.
INT	10 C = INT(A)	Returns largest integer less than or equal to argument.
LOG	10 C = LOG(A)	Returns natural logarithm of argument. Argument must be greater than or equal to zero.
RND	10 C = RND(A)	Generates a random number between zero and one. If 'a' is less then 0, the same random number is produced in each call to rnd. If 'a' = 0, the same sequence of random number is generated each time rnd is called. If 'a' is greater than 0, a new sequence is produced for each call to rnd.
SGN	10 C = SGN(A)	Returns -1 if argument is negative, returns 0 if argument is zero, and returns + 1 if argument is positive.
SIN	10 C = SIN(A)	Returns sin of argument. 'A' must be expressed in radians.
SQR	10 C = SQR(A)	Returns the square root of argument.
TAN	10 C = TAN(A)	Returns tangent of argument. 'A' must be expressed in radians.

Mathematical Functions

Function	**Basic Equivalent**
Secant	SEC(X) = 1 / COS(X)
Cosecant	CSC(X) = 1 / SIN(X)
Cotangent	COT(X) = 1 / TAN(X)
Inverse Sine	ARCSIN(X) = ATN(X / SQR(–X*X + 1))
Inverse Cosine	ARCCOS(X) = ATN(X / SQR(–X*X + 1)) + π/2
Inverse Secant	ARCSEC(X) = ATN(X / SQR(X*X–1))
Inverse Cosecant	ARCCSC(X) = ATN(X / SQR(X*X–1) + (SGN(X) – 1*π/2)
Inverse Cotangent	ARCCOT(X) = ATN(X) + π/2
Hyperbolic Sine	SINH(X) = (EXP(X) – EXP(–X)) / 2
Hyperbolic Cosine	COSH(X) = (EXP(X) + EXP(–X)) / 2
Hyperbolic Tangent	TANH(X) = EXP(–X) / (EXP(X) + EXP(–X)) * 2 + 1
Hyperbolic Secant	SECH(X) = 2 / (EXP(X) + EXP(–X)
Hyperbolic Cosecant	CSCH(X) = 2 / (EXP(X) – EXP(–X)
Hyperbolic Cotangent	COTH(X) = EXP(–X) / (EXP(X) – EXP(–X)) * 2 + 1
Inverse Hyperbolic Sine	ARCSINH(X) = LOG(X + SQR(X*X + 1))
Inverse Hyperbolic Cosine	ARCCOSH(X) = LOG(X / SQR(X*X–1))
Inverse Hyperbolic Tangent	ARCTANH(X) = LOG((1 + X) / (1–X)) / 2
Inverse Hyperbolic Secant	ARCSECH(X) = LOG(SQR(–X*X + 1) + 1/X)
Inverse Hyperbolic Cosecant	ARCCSCH(X) = LOG(X / SQR(X*X–1) + (SGN(X) – 1*π/2)
Inverse Hyperbolic Cotangent	ARCCOTH(X) = LOG(X) + π/2

ST - The Status Variable

Bit	Val	Cassette Read	IEEE	Tape Load/Ver.	Vic/64 RS-232
0-7	0	OK	OK	OK	OK
0	1		time out on write		parity error
1	2		time out on read		framing error
2	4	short block		short block	rec. buffer overrun
3	8	long block		long block	unused
4	16	unrecoverable read error		any mismatch	CTS signal missing
5	32	checksum error		checksum error	unused
6	64	end of file	EOI		DSR signal missing
7	-128	end of tape	device not present	end of tape	break detected

DS & DS$ - Disk Status Variables

DS	Error Description
0	OK, no error exists
1	files scratched response (not an error)
2-19	Unused: can occur, should be ignored
20	read error; block header not found
21	read error; sync character not found
22	read error; data block not present
23	read error; checksum error in data
24	read error; byte decoding error
25	write error; write verify error
26	write protect on
27	read error; checksum error in header
28	write error; data extends into next block
29	disk id mismatch
30	syntax error; general syntax
31	syntax error; invalid command
32	syntax error; command line > 58 chars
33	syntax error; invalid filename
34	syntax error; no filename given
39	syntax error; command file not given
50	record not present
51	overflow in record
52	file too large
60	file open for write
61	file not open
62	file not found
63	file exists
64	file type mismatch
65	no block; t,s is next available block
66	illegal track or sector
67	illegal system track or sector
70	no channels (available)
71	dir error (directory error)
72	disk full or directory full
73	cbm dos v2 (or v2.x for later dos's); power up message, also indicates write attempt with dos mismatch
74	drive not ready
75	format speed error
76	controller error

Printer Control Characters

CHR$ values are sent to print file with Secondary Addr 0 or 1

CHR$	Operation
10	Line Feed
13	'Carriage Return' (automatic Line Feed on CBM printers)
14	Begin double-width character mode
15	End double-width character mode
18	Begin reverse character mode
146	End reverse character mode
19	Set top of page
147	Feed to top of next page
17	Switch to upper/lower case character set
145	Switch to upper case/graphics character set
16	Tab to position in next 2 characters
27	Move to specified dot position
8	Begin dot-programmable graphic mode
26	Repeat graphics data

Table Of Secondary Addresses

Eg. OPEN 4, 4, 7 ; 7 is the Secondary Address on CBM printers that alters line spacing. Once open the new value can be sent. Secondary addresses are not applicable to the VIC 20/Commodore 64 RS-232 routines ('device' 2), keyboard (device 0), screen (device 3), or the CBM 8010 Modem (device 5).

Sec Addr.	I/O Device & Device Number (DV#) Printer 4	Cassette 1 or 2	Vic/64 Cassette 1	Disk 8
0	Print data exactly as received	seq. read	Load & relocate (dflt)	Load, and Dir read
1	Print data according to previously defined format	Write file + end-of-file marker on Close	Load without relocating	Program Save
2	Format Set-up	Write file + eof + end of tape marker on Close	Write file + eof + end of tape marker on Close	R/W channels are 2–14
3	Set number of lines per page for paging			
4	Enable printer format diagnostics			
5	Define a programmable character			
6	Set spacing between lines			
7	Upper/Lower case			
8	ASCII/Graphics			
9	Suppress Diagnostic Message Printing			
10	Reset Printer			
11	Set Uni-Direction			
12	Reset Uni-Direction			
13	Set Condense mode			
14	Reset Condense mode			
15	Set pseudo letter quality			Command Ch.
21	Reset pseudo letter quality			
17	Storing bit image data			
18	Printing bit data previously written			

8032 Control Characters

Most functions can be activated by combinations of simultaneous key depressions, a phenomena of the keyboard hardware. Notice that the CHR$ values of complimentary functions differ by 128.

Function	CHR$	ESC/RVS	Keyboard Combination
BELL	7	G	
GRAPHICS TEXT	142 14	Shift N N	Both Shifts + "
SCROLL DOWN SCROLL UP	153 25	Shift Y Y	Left Shift + TAB + I
SET BOTTOM SET TOP	143 15	Shift O O	Shift + Z + A + L Z + A + L
INSERT LINE DELETE LINE	149 21	Shift U U	Shift + RVS + A + L RVS + A + L
ERASE BEGIN ERASE END	150 22	Shift V V	Shift + TAB + ← + DEL TAB + ← + DEL
SET/CLR TAB TAB	137 9	Shift I I	Shift + TAB TAB

8032 Window POKEs	
TOP:224, T where T = 0 to 24	LEFT:226, L where L = 0 to 79
BOTTOM:225, B where B = T to 24	RIGHT:213, R where R = L to 79

Error Messages

Message	Description
BAD DATA	String data was received from an open file, but the program was expecting numeric data.
BAD SUBSCRIPT	The program was trying to reference an element of an array whose number is outside of the range specified in the DIM statement.
CAN'T CONTINUE	The CONT command will not work, either because the program was never 'RUN', there has been an error, or a line has been edited.
DEVICE NOT PRESENT	The required I/O device was not available for an 'OPEN', 'CLOSE', 'CMD', 'PRINT#', 'INPUT#', or 'GET#'.
DIVISION BY ZERO	Division by zero is a mathematical oddity and not allowed.
EXTRA IGNORED	Too many items of data were typed in response to an input statement. Only the first few items were accepted.
FILE NOT FOUND	If you were looking for a file on tape, an 'end-of-tape' marker was found. If you were looking on a disk, no file with that name exists.
FILE NOT OPEN	The file specified in a 'CLOSE', 'CMD', 'PRINT#', 'INPUT#', or 'GET#', must first be 'OPEN'ed.
FILE OPEN	An attempt was made to OPEN a file using the number of an already open file.
FORMULA TOO COMPLEX	The string expression being evaluated should be split into at least two parts for the system to work with, or a formula has too many parentheses.
ILLEGAL DIRECT	The 'INPUT' statement can only be used within a program, and not in direct mode.
ILLEGAL QUANTITY	A number used as the argument of a function or statement is out of the allowable range.
LOAD	A problem has occured during program LOAD, disk or tape
NEXT WITHOUT FOR	This is caused by either incorrectly nesting loops or having a variable name in a 'NEXT' statement that doesn't correspond with one in a 'FOR' statement.
NOT INPUT FILE	An attempt was made to 'INPUT' or 'GET' data from a file which was specified to be for output only.
NOT OUTPUT FILE	An attempt was made to 'PRINT' data to a file which was specified as input only.
OUT OF DATA	A 'READ' statement was executed but there is no data left unread in a 'DATA' statement.
OUT OF MEMORY	There is no more 'ram' available for program or variables. This may also occur when too many 'FOR' loops have been nested, or when there are too many 'GOSUB's in effect.
OVERFLOW	The result of a computation is larger than the largest number allowed, which is 1.70141884e+38.
REDIM'D ARRAY	An array may only be 'DIM'ensioned once. If an array variable is used before that array is 'DIM'd, an automatic 'DIM' operation is performed on that array setting the number of elements to ten, and any subsequent 'DIM's will cause this error.
REDO FROM START	Character data was typed in during an 'INPUT' statement when numeric data was expected. Just re-type the entry so that it is correct, and the program will continue by itself.
RETURN WITHOUT GOSUB	A 'RETURN' statement was encountered, and no 'GOSUB' command has been issued.
STRING TOO LONG	(except 2.0) A string can contain up to 255 characters.
SYNTAX	A statement or command is unrecognizable. A missing or extra parenthess, misspelled keywords, etc.
TYPE MISMATCH	This error occurs when a number is used in place of a string, or vice-versa.
UNDEF'D FUNCTION	A user defined function was referenced, but it has never been defined using the 'DEF FN' statement.
UNDEF'D STATEMENT	An attempt was made to 'GOTO' or 'GOSUB' or 'RUN' a line number that doesn't exist.
VERIFY	The program on tape or disk does not match the program currently in memory.

B–Series Error Messages

This list is a summary of B–Series error messages that are displayed by PRINTing ERR$(X) where X equals the value down the left column.

X	Message	Explanation
0	?STOP KEY DETECTED	Occurs when doing a KERNAL I/O function and the STOP key is pressed. May occur during LOAD or SAVE (or OPEN, CLOSE, GET#, INPUT#, PRINT# when the cassette tape is moving). CLOSE any open write files to save data.
1	?TOO MANY FILES	Maximum OPEN files is ten.
2	?FILE OPEN	An attempt was made to OPEN or DOPEN a file with a file number already in use.
3	?FILE NOT OPEN	An attempt was made to access a file not previously OPEN or DOPENed
4	?FILE NOT FOUND	The file specified in OPEN or LOAD was not found on the device specified. In the case of tape I/O, and end of tape marker was encountered.
5	?DEVICE NOT PRESENT	An attempt was made to access a device not currently connected or powered-up on the IEEE-488 bus. May happen on OPEN, CLOSE, CMD, INPUT#, GET#, PRINT#. If filename is not specified with OPEN, this error will occur.
6	?NOT INPUT FILE	An attempts was made to read a file originally OPENed for writing.
7	?NOT OUTPUT FILE	An attempts was made to write data to a file originally OPENed for reading. The keyboard cannot be written to.
8	?MISSING FILENAME	All LOADs and SAVEs from the IEEE port (eg. disk) require a filename.

X	Message	Explanation
9	?ILLEGAL DEVICE NUMBER	Occurs if you try to access a device in an illegal manner. For example, LOADing or SAVEing from/to the keyboard, screen, or RS-232.
10	?ARE YOU SURE	Confirmation prompt for BACKUP, SCRATCH, and HEADER. It is not an error message and occurs only in direct mode, not during BASIC program execution.
11	?BAD DISK	Media failure on HEADER command.
12	<return> READY. <return>	This Is Not An Error Message. This message lets you know that your system is ready to use.
13	<space> IN <space>	Not An Error Message. Used to indicate which line an error has occurred "in".
14	?BREAK	This occurs when the STOP key is pressed during BASIC execution. CONT can be used to restart the program.
15	?EXTRA IGNORED	Too many items of data or separators were entered in response to an INPUT statement.
16	?REDO FROM START	This diagnostic message occurs when a numeric variable is used with INPUT and non-numeric data is received. INPUT continues to function until acceptable data has been received.
17	Last Evaluated Number	This Is Not An Error Message. This is the last value that has been processed through the numerical output buffer. (eg. print 100/10 : print ERR$(17) ...will print 10 both times.
18	"MORE" <return>	This Is Not An Error Message. Prints "MORE" and carriage return.
19	Power On Message	This Is Not An Error Message. Prints: *** COMMODORE BASIC 128, B4.0 *** *** COMMODORE BASIC 256, B4.0 ***
20	?NEXT WITHOUT FOR	Either a NEXT is improperly nested or the variable in a NEXT statement corresponds to no previously executed FOR statement.
21	?SYNTAX	BASIC cannot recognize the statement you have typed. Caused by such things as missing parenthesis, illegal characters, incorrect punctuation, misspelled keyword.
22	?RETURN WITHOUT GOSUB	A RETURN statement was encountered with noprevious GOSUB.
23	?OUT OF DATA	An attempt was made to READ data from a DATA statement but no data exists or the program has already read them all.
24	?ILLEGAL QUANTITY	Occurs when a function is accessed with a parameter out of range caused by: 1. A matrix subscript out of range (0 < X < 32767) 2. LOG (negative or zero argument) 3. SQR (negative argument) 4. A↑B where A<0 and B not integer. 5. Call of USR before a machine language subroutine has been patched in. 6. Use of string functions MID$, LEFT$, RIGHT$, with length parameters out of range. 7. Index on...GOTO out of range. 8. Addressof PEEK, POKE, WAIT or SYS out of range. 9. Byte parameters of WAIT, POKE, TAB and SPC out of range.
25	?OVERFLOW	Numbers resulting from computations or input that are greater than 1.70141184E+38 or less than 2.93873587E-39.
26	?OUT OF MEMORY	BASIC text space, or Variables space, or Arrays memory space has been completely filled
27	?UNDEFINED STATEMENT	A GOTO, GOSUB, or THEN has been executed with a line number that does not exist.
28	?BAD SUBSCRIPT	An attempt was made to reference an array element which is outside the dimensions specified in the DIM statement.
29	?REDIM'D ARRAY	An attempt was made to define an array using a variable already used in an array.
30	?DIVISION BY ZERO	Illegal divide. Message is followed by the line number – list and check variables.
31	?ILLEGAL DIRECT	INPUT, INPUT#, GET, GET#, and DEF cannot be used in direct mode.
32	?TYPE MISMATCH	An arithmetic operation has been given non-numeric data, or a string operation has been numeric data.
33	?STRING TOO LONG	Maximum string length is 255 characters. This error will also occur if INPUT# receives more than 80 characters without a carriage return (ie. BASIC input buffer is 80 bytes long).
34	?FILE DATA	Occurs when a numeric variable is used with INPUT# and non-numeric data is received.
35	?FORMULA TOO COMPLEX	BASIC has run out of temporary pointers to keep track of substrings in evaluating a string expression. Break the expression into two smaller parts to cure the problem.
37	?UNDEFINED FUNCTION	Reference was made to a user defined function which had never been defined with DEF.
38	?LOAD ERROR	Cassette tape only. To improve tape reliability, programs are recorded twice with SAVE. This error will occur if LOAD finds recording errors in corresponding positions of both recordings. If more than 31 errors are detected in the first pass, LOAD will not attempt to read the second.
39	?VERIFY ERROR	A VERIFY operation did not match the contents of file with the contents of memory. Re-SAVE your program on another disk or tape.
40	?OUT OF STACK	Too many open FOR...NEXT loops or too many GOSUB calls.
41	?UNABLE TO RESUME	Resume will not operate after a fatal error.
42	?UNABLE TO DISPOSE	All of the DISPOSE type items have been disposed of or none exist.
43	?OUT OF TEXT	A LOAD or DLOAD has attempted to bring in a file larger than 64K. This error will not occur when using the BLOAD command.

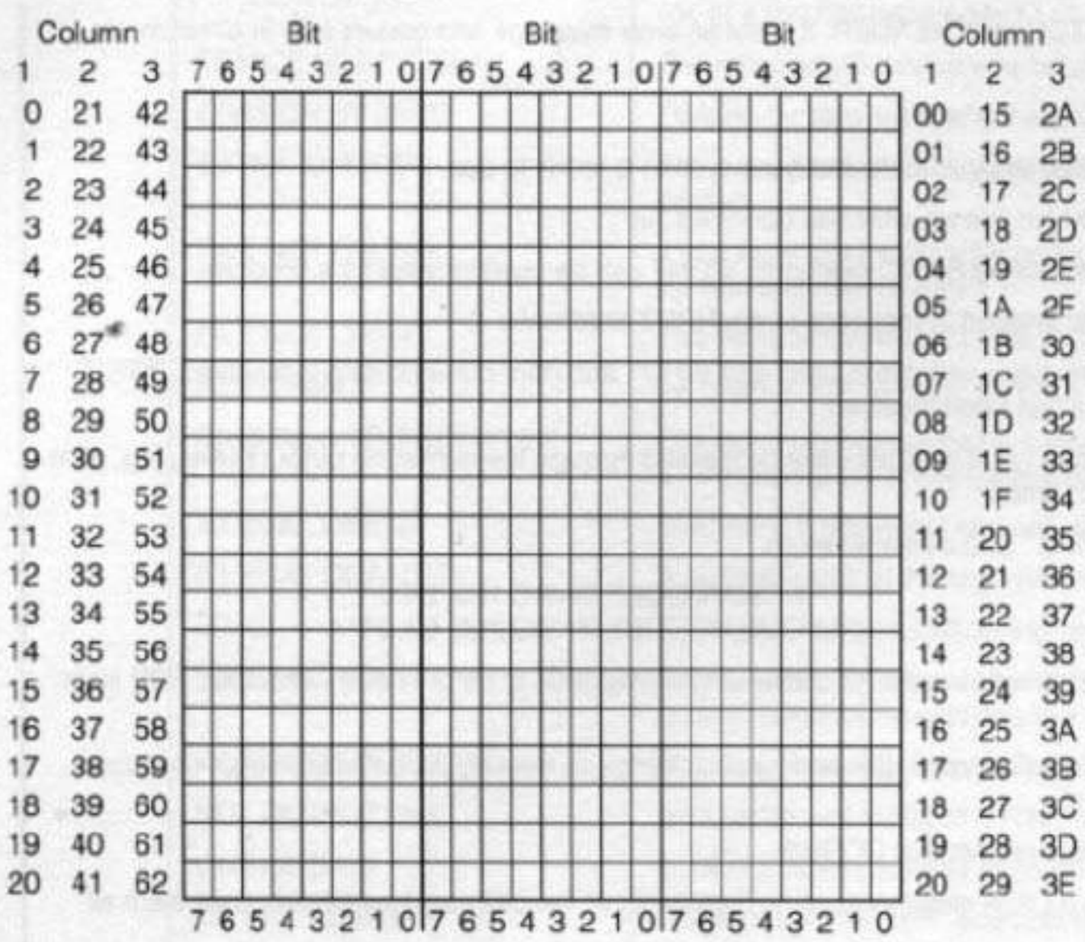
Sprite Design

Sprite Colour #2 _____: POKE 53285, _____
Sprite Colour #3 _____: POKE 53286, _____
Sprite Enable: POKE 53269, PEEK(53269) OR 2↑Sprite#
POKE Sprite X-Expand: POKE 53264, PEEK(53264) OR 2↑Sprite#
Sprite Y-Expand: POKE 53271, PEEK(53271) OR 2↑Sprite#
Background Priority: POKE 53275, PEEK(53275) OR 2↑Sprite#

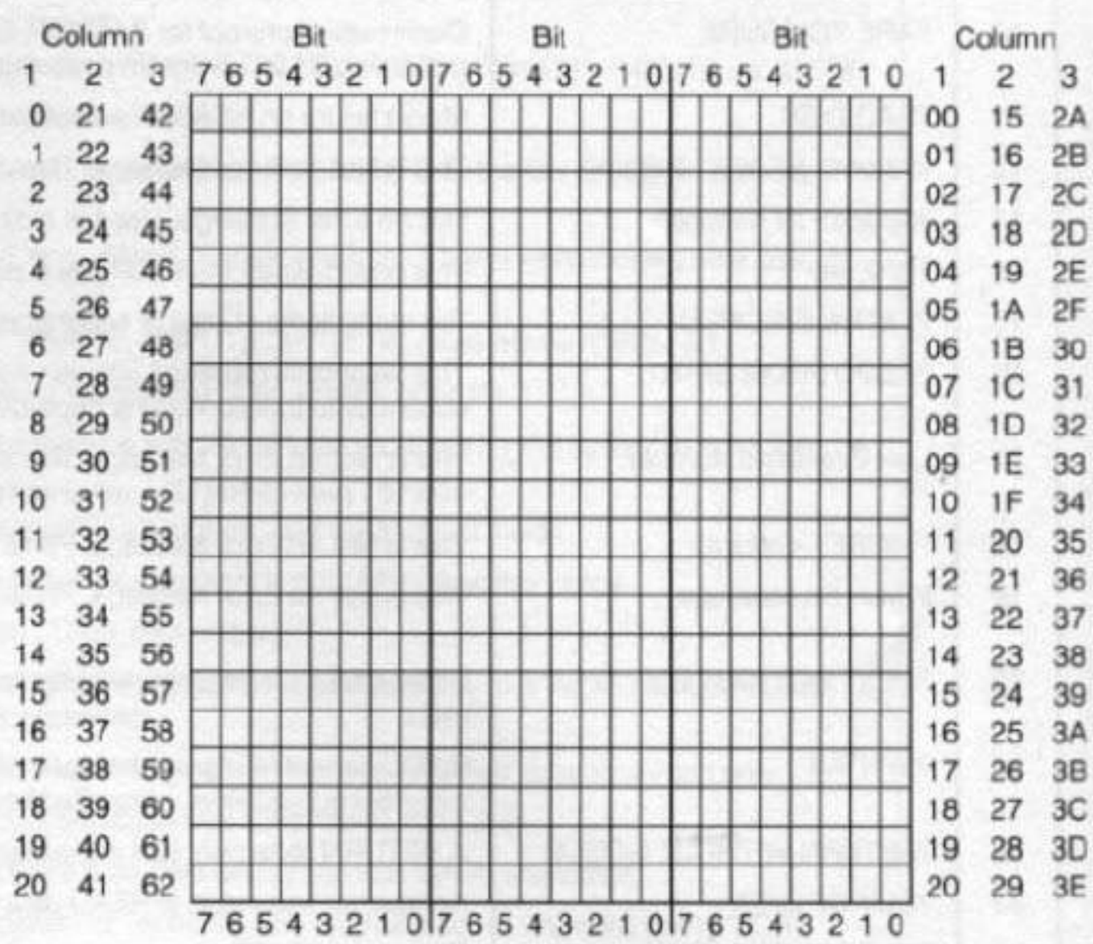
Sprite Multi Colour Mode: POKE 53276, PEEK(53276) OR 2↑Sprite#

Multi Colour Mode Bit Pairs

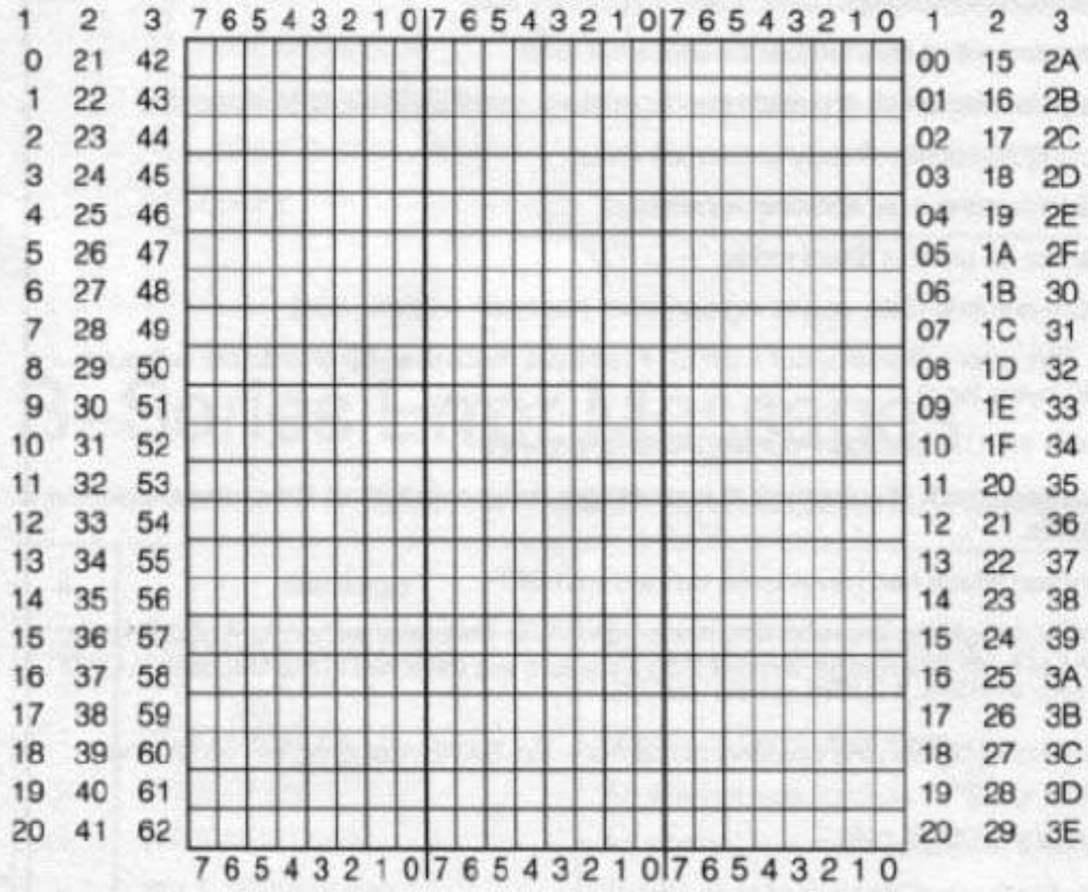
Background Colour, PEEK(53281), Use: 00
Sprite Colour Use: 01
Sprite Colour #2 Use: 10
Sprite Colour #3 Use: 11



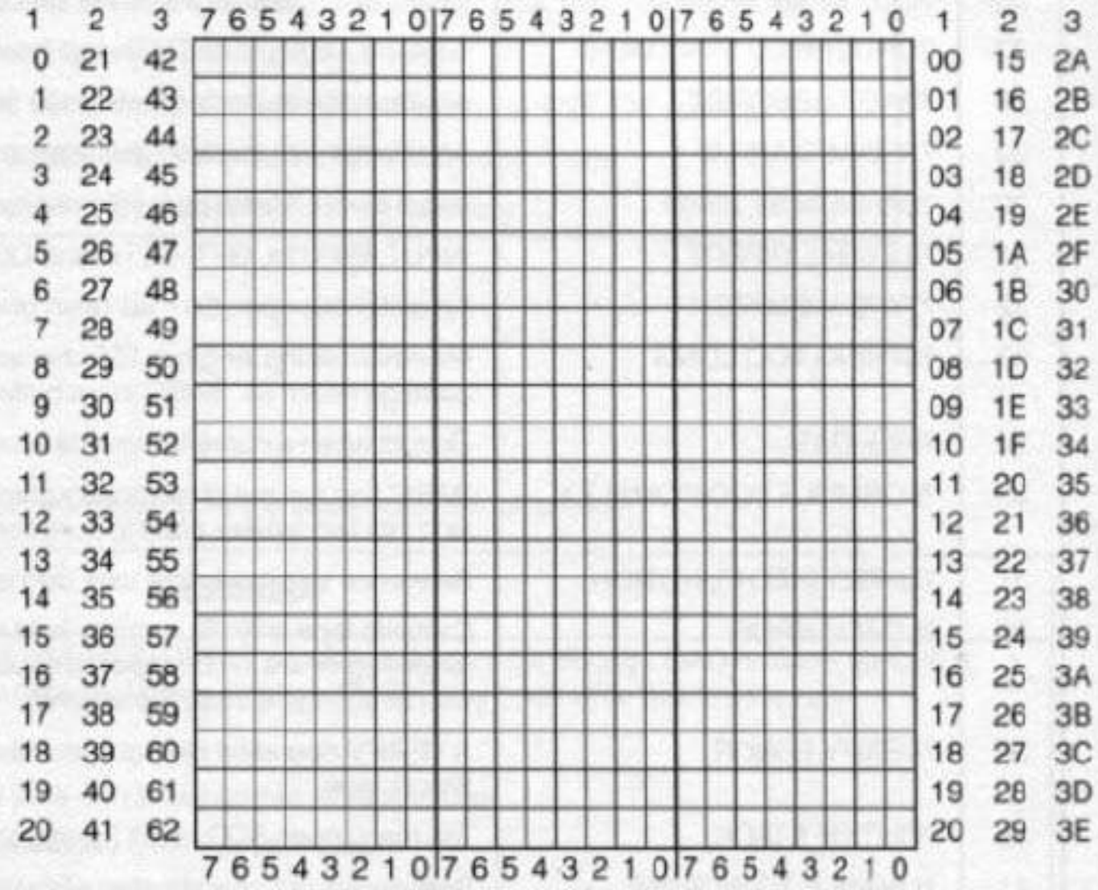
Sprite # _____ (0–7)
Pointer: POKE 2040 + Sprite#, _____
Sprite Colour: _____ : POKE 53287 + Sprite#, _____
X-Position: POKE 53248 + Sprite#, X Position
Y-Position: POKE 53249 + Sprite#, Y Position



Sprite # _____ (0–7)
Pointer: POKE 2040 + Sprite#, _____
Sprite Colour: _____ : POKE 53287 + Sprite#, _____
X-Position: POKE 53248 + Sprite#, X Position
Y-Position: POKE 53249 + Sprite#, Y Position



Sprite # _____ (0–7)
Pointer: POKE 2040 + Sprite#, _____
Sprite Colour: _____ : POKE 53287 + Sprite#, _____
X-Position: POKE 53248 + Sprite#, X Position
Y-Position: POKE 53249 + Sprite#, Y Position



Sprite # _____ (0–7)
Pointer: POKE 2040 + Sprite#, _____
Sprite Colour: _____ : POKE 53287 + Sprite#, _____
X-Position: POKE 53248 + Sprite#, X Position
Y-Position: POKE 53249 + Sprite#, Y Position

Character Design

	Bit 7	6	5	4	3	2	1	0
0								
1								
2								
3								
4								
5								
6								
7								

Character # ______

	Bit 7	6	5	4	3	2	1	0
0								
1								
2								
3								
4								
5								
6								
7								

Character # ______

	Bit 7	6	5	4	3	2	1	0
0								
1								
2								
3								
4								
5								
6								
7								

Character # ______

Screen Design

40 Column PET/CBM Screen Map

32768

1	8000	2768	2769	2770	2771	2772	2773	2774	2775	2776	2777	2778	2779	2780	2781	2782	2783	2784	2785	2786	2787	2788	2789	2790	2791	2792	2793	2794	2795	2796	2797	2798	2799	2800	2801	2802	2803	2804	2805	2806	2807
2	8028	2808	2809	2810	2811	2812	2813	2814	2815	2816	2817	2818	2819	2820	2821	2822	2823	2824	2825	2826	2827	2828	2829	2830	2831	2832	2833	2834	2835	2836	2837	2838	2839	2840	2841	2842	2843	2844	2845	2846	2847
3	8050	2848	2849	2850	2851	2852	2853	2854	2855	2856	2857	2858	2859	2860	2861	2862	2863	2864	2865	2866	2867	2868	2869	2870	2871	2872	2873	2874	2875	2876	2877	2878	2879	2880	2881	2882	2883	2884	2885	2886	2887
4	8078	2888	2889	2890	2891	2892	2893	2894	2895	2896	2897	2898	2899	2900	2901	2902	2903	2904	2905	2906	2907	2908	2909	2910	2911	2912	2913	2914	2915	2916	2917	2918	2919	2920	2921	2922	2923	2924	2925	2926	2927
5	80A0	2928	2929	2930	2931	2932	2933	2934	2935	2936	2937	2938	2939	2940	2941	2942	2943	2944	2945	2946	2947	2948	2949	2950	2951	2952	2953	2954	2955	2956	2957	2958	2959	2960	2961	2962	2963	2964	2965	2966	2967
6	80C8	2968	2969	2970	2971	2972	2973	2974	2975	2976	2977	2978	2979	2980	2981	2982	2983	2984	2985	2986	2987	2988	2989	2990	2991	2992	2993	2994	2995	2996	2997	2998	2999	3000	3001	3002	3003	3004	3005	3006	3007
7	80F0	3008	3009	3010	3011	3012	3013	3014	3015	3016	3017	3018	3019	3020	3021	3022	3023	3024	3025	3026	3027	3028	3029	3030	3031	3032	3033	3034	3035	3036	3037	3038	3039	3040	3041	3042	3043	3044	3045	3046	3047
8	8118	3048	3049	3050	3051	3052	3053	3054	3055	3056	3057	3058	3059	3060	3061	3062	3063	3064	3065	3066	3067	3068	3069	3070	3071	3072	3073	3074	3075	3076	3077	3078	3079	3080	3081	3082	3083	3084	3085	3086	3087
9	8140	3088	3089	3090	3091	3092	3093	3094	3095	3096	3097	3098	3099	3100	3101	3102	3103	3104	3105	3106	3107	3108	3109	3110	3111	3112	3113	3114	3115	3116	3117	3118	3119	3120	3121	3122	3123	3124	3125	3126	3127
10	8168	3128	3129	3130	3131	3132	3133	3134	3135	3136	3137	3138	3139	3140	3141	3142	3143	3144	3145	3146	3147	3148	3149	3150	3151	3152	3153	3154	3155	3156	3157	3158	3159	3160	3161	3162	3163	3164	3165	3166	3167
11	8190	3168	3169	3170	3171	3172	3173	3174	3175	3176	3177	3178	3179	3180	3181	3182	3183	3184	3185	3186	3187	3188	3189	3190	3191	3192	3193	3194	3195	3196	3197	3198	3199	3200	3201	3202	3203	3204	3205	3206	3207
12	81B8	3208	3209	3210	3211	3212	3213	3214	3215	3216	3217	3218	3219	3220	3221	3222	3223	3224	3225	3226	3227	3228	3229	3230	3231	3232	3233	3234	3235	3236	3237	3238	3239	3240	3241	3242	3243	3244	3245	3246	3247
13	81E0	3248	3249	3250	3251	3252	3253	3254	3255	3256	3257	3258	3259	3260	3261	3262	3263	3264	3265	3266	3267	3268	3269	3270	3271	3272	3273	3274	3275	3276	3277	3278	3279	3280	3281	3282	3283	3284	3285	3286	3287
14	8208	3288	3289	3290	3291	3292	3293	3294	3295	3296	3297	3298	3299	3300	3301	3302	3303	3304	3305	3306	3307	3308	3309	3310	3311	3312	3313	3314	3315	3316	3317	3318	3319	3320	3321	3322	3323	3324	3325	3326	3327
15	8230	3328	3329	3330	3331	3332	3333	3334	3335	3336	3337	3338	3339	3340	3341	3342	3343	3344	3345	3346	3347	3348	3349	3350	3351	3352	3353	3354	3355	3356	3357	3358	3359	3360	3361	3362	3363	3364	3365	3366	3367
16	8258	3368	3369	3370	3371	3372	3373	3374	3375	3376	3377	3378	3379	3380	3381	3382	3383	3384	3385	3386	3387	3388	3389	3390	3391	3392	3393	3394	3395	3396	3397	3398	3399	3400	3401	3402	3403	3404	3405	3406	3407
17	8280	3408	3409	3410	3411	3412	3413	3414	3415	3416	3417	3418	3419	3420	3421	3422	3423	3424	3425	3426	3427	3428	3429	3430	3431	3432	3433	3434	3435	3436	3437	3438	3439	3440	3441	3442	3443	3444	3445	3446	3447
18	82A8	3448	3449	3450	3451	3452	3453	3454	3455	3456	3457	3458	3459	3460	3461	3462	3463	3464	3465	3466	3467	3468	3469	3470	3471	3472	3473	3474	3475	3476	3477	3478	3479	3480	3481	3482	3483	3484	3485	3486	3487
19	82D0	3488	3489	3490	3491	3492	3493	3494	3495	3496	3497	3498	3499	3500	3501	3502	3503	3504	3505	3506	3507	3508	3509	3510	3511	3512	3513	3514	3515	3516	3517	3518	3519	3520	3521	3522	3523	3524	3525	3526	3527
20	82F8	3528	3529	3530	3531	3532	3533	3534	3535	3536	3537	3538	3539	3540	3541	3542	3543	3544	3545	3546	3547	3548	3549	3550	3551	3552	3553	3554	3555	3556	3557	3558	3559	3560	3561	3562	3563	3564	3565	3566	3567
21	8320	3568	3569	3570	3571	3572	3573	3574	3575	3576	3577	3578	3579	3580	3581	3582	3583	3584	3585	3586	3587	3588	3589	3590	3591	3592	3593	3594	3595	3596	3597	3598	3599	3600	3601	3602	3603	3604	3605	3606	3607
22	8348	3608	3609	3610	3611	3612	3613	3614	3615	3616	3617	3618	3619	3620	3621	3622	3623	3624	3625	3626	3627	3628	3629	3630	3631	3632	3633	3634	3635	3636	3637	3638	3639	3640	3641	3642	3643	3644	3645	3646	3647
23	8370	3648	3649	3650	3651	3652	3653	3654	3655	3656	3657	3658	3659	3660	3661	3662	3663	3664	3665	3666	3667	3668	3669	3670	3671	3672	3673	3674	3675	3676	3677	3678	3679	3680	3681	3682	3683	3684	3685	3686	3687
24	8398	3688	3689	3690	3691	3692	3693	3694	3695	3696	3697	3698	3699	3700	3701	3702	3703	3704	3705	3706	3707	3708	3709	3710	3711	3712	3713	3714	3715	3716	3717	3718	3719	3720	3721	3722	3723	3724	3725	3726	3727
25	83C0	3728	3729	3730	3731	3732	3733	3734	3735	3736	3737	3738	3739	3740	3741	3742	3743	3744	3745	3746	3747	3748	3749	3750	3751	3752	3753	3754	3755	3756	3757	3758	3759	3760	3761	3762	3763	3764	3765	3766	3767

33767

VIC 20 Screen Map (with expansion memory at $2000)

		4096																					
1	1000	4096	4097	4098	4099	4100	4101	4102	4103	4104	4105	4106	4107	4108	4109	4110	4111	4112	4113	4114	4115	4116	4117
2	1016	4118	4119	4120	4121	4122	4123	4124	4125	4126	4127	4128	4129	4130	4131	4132	4133	4134	4135	4136	4137	4138	4139
3	102C	4140	4141	4142	4143	4144	4145	4146	4147	4148	4149	4150	4151	4152	4153	4154	4155	4156	4157	4158	4159	4160	4161
4	1042	4162	4163	4164	4165	4166	4167	4168	4169	4170	4171	4172	4173	4174	4175	4176	4177	4178	4179	4180	4181	4182	4183
5	1058	4184	4185	4186	4187	4188	4189	4190	4191	4192	4193	4194	4195	4196	4197	4198	4199	4200	4201	4202	4203	4204	4205
6	106E	4206	4207	4208	4209	4210	4211	4212	4213	4214	4215	4216	4217	4218	4219	4220	4221	4222	4223	4224	4225	4226	4227
7	1084	4228	4229	4230	4231	4232	4233	4234	4235	4236	4237	4238	4239	4240	4241	4242	4243	4244	4245	4246	4247	4248	4249
8	109A	4250	4251	4252	4253	4254	4255	4256	4257	4258	4259	4260	4261	4262	4263	4264	4265	4266	4267	4268	4269	4270	4271
9	1060	4272	4273	4274	4275	4276	4277	4278	4279	4280	4281	4282	4283	4284	4285	4286	4287	4288	4289	4290	4291	4292	4293
10	10C6	4294	4295	4296	4297	4298	4299	4300	4301	4302	4303	4304	4305	4306	4307	4308	4309	4310	4311	4312	4313	4314	4315
11	10DC	4316	4317	4318	4319	4320	4321	4322	4323	4324	4325	4326	4327	4328	4329	4330	4331	4332	4333	4334	4335	4336	4337
12	10F2	4338	4339	4340	4341	4342	4343	4344	4345	4346	4347	4348	4349	4350	4351	4352	4353	4354	4355	4356	4357	4358	4359
13	1108	4360	4361	4362	4363	4364	4365	4366	4367	4368	4369	4370	4371	4372	4373	4374	4375	4376	4377	4378	4379	4380	4381
14	111E	4382	4383	4384	4385	4386	4387	4388	4389	4390	4391	4392	4393	4394	4395	4396	4397	4398	4399	4400	4401	4402	4403
15	1134	4404	4405	4406	4407	4408	4409	4410	4411	4412	4413	4414	4415	4416	4417	4418	4419	4420	4421	4422	4423	4424	4425
16	114A	4426	4427	4428	4429	4430	4431	4432	4433	4434	4435	4436	4437	4438	4439	4440	4441	4442	4443	4444	4445	4446	4447
17	1160	4448	4449	4450	4451	4452	4453	4454	4455	4456	4457	4458	4459	4460	4461	4462	4463	4464	4465	4466	4467	4468	4469
18	1176	4470	4471	4472	4473	4474	4475	4476	4477	4478	4479	4480	4481	4482	4483	4484	4485	4486	4487	4488	4489	4490	4491
19	118C	4492	4493	4494	4495	4496	4497	4498	4499	4500	4501	4502	4503	4504	4505	4506	4507	4508	4509	4510	4511	4512	4513
20	11A2	4514	4515	4516	4517	4518	4519	4520	4521	4522	4523	4524	4525	4526	4527	4528	4529	4530	4531	4532	4533	4534	4535
21	1168	4536	4537	4538	4539	4540	4541	4542	4543	4544	4545	4546	4547	4548	4549	4550	4551	4552	4553	4554	4555	4556	4557
22	11CE	4558	4559	4560	4561	4562	4563	4564	4565	4566	4567	4568	4569	4570	4571	4572	4573	4574	4575	4576	4577	4578	4579
23	11E4	4580	4581	4582	4583	4584	4585	4586	4587	4588	4589	4590	4591	4592	4593	4594	4595	4596	4597	4598	4599	4600	4601

4601

VIC 20 Colour Table Map (with expansion memory)

		37888																					
1	9400	7888	7889	7890	7891	7892	7893	7894	7895	7896	7897	7898	7899	7900	7901	7902	7903	7904	7905	7906	7907	7908	7909
2	9416	7910	7911	7912	7913	7914	7915	7916	7917	7918	7919	7920	7921	7922	7923	7924	7925	7926	7927	7928	7929	7930	7931
3	942C	7932	7933	7934	7935	7936	7937	7938	7939	7940	7941	7942	7943	7944	7945	7946	7947	7948	7949	7950	7951	7952	7953
4	9442	7954	7955	7956	7957	7958	7959	7960	7961	7962	7963	7964	7965	7966	7967	7968	7969	7970	7971	7972	7973	7974	7975
5	9458	7976	7977	7978	7979	7980	7981	7982	7983	7984	7985	7986	7987	7988	7989	7990	7991	7992	7993	7994	7995	7996	7997
6	946E	7998	7999	8000	8001	8002	8003	8004	8005	8006	8007	8008	8009	8010	8011	8012	8013	8014	8015	8016	8017	8018	8019
7	9484	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	8030	8031	8032	8033	8034	8035	8036	8037	8038	8039	8040	8041
8	949A	8042	8043	8044	8045	8046	8047	8048	8049	8050	8051	8052	8053	8054	8055	8056	8057	8058	8059	8060	8061	8062	8063
9	9450	8064	8065	8066	8067	8068	8069	8070	8071	8072	8073	8074	8075	8076	8077	8078	8079	8080	8081	8082	8083	8084	8085
10	94C6	8086	8087	8088	8089	8090	8091	8092	8093	8094	8095	8096	8097	8098	8099	8100	8101	8102	8103	8104	8105	8106	8107
11	94DC	8108	8109	8110	8111	8112	8113	8114	8115	8116	8117	8118	8119	8120	8121	8122	8123	8124	8125	8126	8127	8128	8129
12	94F2	8130	8131	8132	8133	8134	8135	8136	8137	8138	8139	8140	8141	8142	8143	8144	8145	8146	8147	8148	8149	8150	8151
13	9508	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163	8164	8165	8166	8167	8168	8169	8170	8171	8172	8173
14	951E	8174	8175	8176	8177	8178	8179	8180	8181	8182	8183	8184	8185	8186	8187	8188	8189	8190	8191	8192	8193	8194	8195
15	9534	8196	8197	8198	8199	8200	8201	8202	8203	8204	8205	8206	8207	8208	8209	8210	8211	8212	8213	8214	8215	8216	8217
16	954A	8218	8219	8220	8221	8222	8223	8224	8225	8226	8227	8228	8229	8230	8231	8232	8233	8234	8235	8236	8237	8238	8239
17	9560	8240	8241	8242	8243	8244	8245	8246	8247	8248	8249	8250	8251	8252	8253	8254	8255	8256	8257	8258	8259	8260	8261
18	9576	8262	8263	8264	8265	8266	8267	8268	8269	8270	8271	8272	8273	8274	8275	8276	8277	8278	8279	8280	8281	8282	8283
19	958C	8284	8285	8286	8287	8288	8289	8290	8291	8292	8293	8294	8295	8296	8297	8298	8299	8300	8301	8302	8303	8304	8305
20	95A2	8306	8307	8308	8309	8310	8311	8312	8313	8314	8315	8316	8317	8318	8319	8320	8321	8322	8323	8324	8325	8326	8327
21	9568	8328	8329	8330	8331	8332	8333	8334	8335	8336	8337	8338	8339	8340	8341	8342	8343	8344	8345	8346	8347	8348	8349
22	95CE	8350	8351	8352	8353	8354	8355	8356	8357	8358	8359	8360	8361	8362	8363	8364	8365	8366	8367	8368	8369	8370	8371
23	95E4	8372	8373	8374	8375	8376	8377	8378	8379	8380	8381	8382	8383	8384	8385	8386	8387	8388	8389	8390	8391	8392	8393

38393

VIC 20 Screen Map (without expansion memory)

7680

1	1E00	7680	7681	7682	7683	7684	7685	7686	7687	7688	7689	7690	7691	7692	7693	7694	7695	7696	7697	7698	7699	7700	7701
2	1E16	7702	7703	7704	7705	7706	7707	7708	7709	7710	7711	7712	7713	7714	7715	7716	7717	7718	7719	7720	7721	7722	7723
3	1E2C	7724	7725	7726	7727	7728	7729	7730	7731	7732	7733	7734	7735	7736	7737	7738	7739	7740	7741	7742	7743	7744	7745
4	1E42	7746	7747	7748	7749	7750	7751	7752	7753	7754	7755	7756	7757	7758	7759	7760	7761	7762	7763	7764	7765	7766	7767
5	1E58	7768	7769	7770	7771	7772	7773	7774	7775	7776	7777	7778	7779	7780	7781	7782	7783	7784	7785	7786	7787	7788	7789
6	1E6E	7790	7791	7792	7793	7794	7795	7796	7797	7798	7799	7800	7801	7802	7803	7804	7805	7806	7807	7808	7809	7810	7811
7	1E84	7812	7813	7814	7815	7816	7817	7818	7819	7820	7821	7822	7823	7824	7825	7826	7827	7828	7829	7830	7831	7832	7833
8	1E9A	7834	7835	7836	7837	7838	7839	7840	7841	7842	7843	7844	7845	7846	7847	7848	7849	7850	7851	7852	7853	7854	7855
9	1E60	7856	7857	7858	7859	7860	7861	7862	7863	7864	7865	7866	7867	7868	7869	7870	7871	7872	7873	7874	7875	7876	7877
10	1EC6	7878	7879	7880	7881	7882	7883	7884	7885	7886	7887	7888	7889	7890	7891	7892	7893	7894	7895	7896	7897	7898	7899
11	1EDC	7900	7901	7902	7903	7904	7905	7906	7907	7908	7909	7910	7911	7912	7913	7914	7915	7916	7917	7918	7919	7920	7921
12	1EF2	7922	7923	7924	7925	7926	7927	7928	7929	7930	7931	7932	7933	7934	7935	7936	7937	7938	7939	7940	7941	7942	7943
13	1F08	7944	7945	7946	7947	7948	7949	7950	7951	7952	7953	7954	7955	7956	7957	7958	7959	7960	7961	7962	7963	7964	7965
14	1F1E	7966	7967	7968	7969	7970	7971	7972	7973	7974	7975	7976	7977	7978	7979	7980	7981	7982	7983	7984	7985	7986	7987
15	1F34	7988	7989	7990	7991	7992	7993	7994	7995	7996	7997	7998	7999	8000	8001	8002	8003	8004	8005	8006	8007	8008	8009
16	1F4A	8010	8011	8012	8013	8014	8015	8016	8017	8018	8019	8020	8021	8022	8023	8024	8025	8026	8027	8028	8029	8030	8031
17	1F60	8032	8033	8034	8035	8036	8037	8038	8039	8040	8041	8042	8043	8044	8045	8046	8047	8048	8049	8050	8051	8052	8053
18	1F76	8054	8055	8056	8057	8058	8059	8060	8061	8062	8063	8064	8065	8066	8067	8068	8069	8070	8071	8072	8073	8074	8075
19	1F8C	8076	8077	8078	8079	8080	8081	8082	8083	8084	8085	8086	8087	8088	8089	8090	8091	8092	8093	8094	8095	8096	8097
20	1FA2	8098	8099	8100	8101	8102	8103	8104	8105	8106	8107	8108	8109	8110	8111	8112	8113	8114	8115	8116	8117	8118	8119
21	1FB8	8120	8121	8122	8123	8124	8125	8126	8127	8128	8129	8130	8131	8132	8133	8134	8135	8136	8137	8138	8139	8140	8141
22	1FCE	8142	8143	8144	8145	8146	8147	8148	8149	8150	8151	8152	8153	8154	8155	8156	8157	8158	8159	8160	8161	8162	8163
23	1FE4	8164	8165	8166	8167	8168	8169	8170	8171	8172	8173	8174	8175	8176	8177	8178	8179	8180	8181	8182	8183	8184	8185

8185

VIC 20 Colour Table Map (without expansion memory)

38400

1	9600	8400	8401	8402	8403	8404	8405	8406	8407	8408	8409	8410	8411	8412	8413	8414	8415	8416	8417	8418	8419	8420	8421
2	9616	8422	8423	8424	8425	8426	8427	8428	8429	8430	8431	8432	8433	8434	8435	8436	8437	8438	8439	8440	8441	8442	8443
3	962C	8444	8445	8446	8447	8448	8449	8450	8451	8452	8453	8454	8455	8456	8457	8458	8459	8460	8461	8462	8463	8464	8465
4	9642	8466	8467	8468	8469	8470	8471	8472	8473	8474	8475	8476	8477	8478	8479	8480	8481	8482	8483	8484	8485	8486	8487
5	9658	8488	8489	8490	8491	8492	8493	8494	8495	8496	8497	8498	8499	8500	8501	8502	8503	8504	8505	8506	8507	8508	8509
6	966E	8510	8511	8512	8513	8514	8515	8516	8517	8518	8519	8520	8521	8522	8523	8524	8525	8526	8527	8528	8529	8530	8531
7	9684	8532	8533	8534	8535	8536	8537	8538	8539	8540	8541	8542	8543	8544	8545	8546	8547	8548	8549	8550	8551	8552	8553
8	969A	8554	8555	8556	8557	8558	8559	8560	8561	8562	8563	8564	8565	8566	8567	8568	8569	8570	8571	8572	8573	8574	8575
9	9660	8576	8577	8578	8579	8580	8581	8582	8583	8584	8585	8586	8587	8588	8589	8590	8591	8592	8593	8594	8595	8596	8597
10	96C6	8598	8599	8600	8601	8602	8603	8604	8605	8606	8607	8608	8609	8610	8611	8612	8613	8614	8615	8616	8617	8618	8619
11	96DC	8620	8621	8622	8623	8624	8625	8626	8627	8628	8629	8630	8631	8632	8633	8634	8635	8636	8637	8638	8639	8640	8641
12	96F2	8642	8643	8644	8645	8646	8647	8648	8649	8650	8651	8652	8653	8654	8655	8656	8657	8658	8659	8660	8661	8662	8663
13	9708	8664	8665	8666	8667	8668	8669	8670	8671	8672	8673	8674	8675	8676	8677	8678	8679	8680	8681	8682	8683	8684	8685
14	971E	8686	8687	8688	8689	8690	8691	8692	8693	8694	8695	8696	8697	8698	8699	8700	8701	8702	8703	8704	8705	8706	8707
15	9734	8708	8709	8710	8711	8712	8713	8714	8715	8716	8717	8718	8719	8720	8721	8722	8723	8724	8725	8726	8727	8728	8729
16	974A	8730	8731	8732	8733	8734	8735	8736	8737	8738	8739	8740	8741	8742	8743	8744	8745	8746	8747	8748	8749	8750	8751
17	9760	8752	8753	8754	8755	8756	8757	8758	8759	8760	8761	8762	8763	8764	8765	8766	8767	8768	8769	8770	8771	8772	8773
18	9776	8774	8775	8776	8777	8778	8779	8780	8781	8782	8783	8784	8785	8786	8787	8788	8789	8790	8791	8792	8793	8794	8795
19	978C	8796	8797	8798	8799	8800	8801	8802	8803	8804	8805	8806	8807	8808	8809	8810	8811	8812	8813	8814	8815	8816	8817
20	97A2	8818	8819	8820	8821	8822	8823	8824	8825	8826	8827	8828	8829	8830	8831	8832	8833	8834	8835	8836	8837	8838	8839
21	97B8	8840	8841	8842	8843	8844	8845	8846	8847	8848	8849	8850	8851	8852	8853	8854	8855	8856	8857	8858	8859	8860	8861
22	97CE	8862	8863	8864	8865	8866	8867	8868	8869	8870	8871	8872	8873	8874	8875	8876	8877	8878	8879	8880	8881	8882	8883
23	97E4	8884	8885	8886	8887	8888	8889	8890	8891	8892	8893	8894	8895	8896	8897	8898	8899	8900	8901	8902	8903	8904	8905

38905

Commodore 64 Screen Map (default)

		1024																																				
1	0400	1024	1025	1026	1027	1028	1029	1030	1031	1032	1033	1034	1035	1036	1037	1038	1039	1040	1041	1042	1043	1044	1045	1046	1047	1048	1049	1050	1051	1052	1053	1054	1055	1056	1057	1058	1059	1060	1061	1062	1063
2	0428	1064	1065	1066	1067	1068	1069	1070	1071	1072	1073	1074	1075	1076	1077	1078	1079	1080	1081	1082	1083	1084	1085	1086	1087	1088	1089	1090	1091	1092	1093	1094	1095	1096	1097	1098	1099	1100	1101	1102	1103
3	0450	1104	1105	1106	1107	1108	1109	1110	1111	1112	1113	1114	1115	1116	1117	1118	1119	1120	1121	1122	1123	1124	1125	1126	1127	1128	1129	1130	1131	1132	1133	1134	1135	1136	1137	1138	1139	1140	1141	1142	1143
4	0478	1144	1145	1146	1147	1148	1149	1150	1151	1152	1153	1154	1155	1156	1157	1158	1159	1160	1161	1162	1163	1164	1165	1166	1167	1168	1169	1170	1171	1172	1173	1174	1175	1176	1177	1178	1179	1180	1181	1182	1183
5	04A0	1184	1185	1186	1187	1188	1189	1190	1191	1192	1193	1194	1195	1196	1197	1198	1199	1200	1201	1202	1203	1204	1205	1206	1207	1208	1209	1210	1211	1212	1213	1214	1215	1216	1217	1218	1219	1220	1221	1222	1223
6	04C8	1224	1225	1226	1227	1228	1229	1230	1231	1232	1233	1234	1235	1236	1237	1238	1239	1240	1241	1242	1243	1244	1245	1246	1247	1248	1249	1250	1251	1252	1253	1254	1255	1256	1257	1258	1259	1260	1261	1262	1263
7	04F0	1264	1265	1266	1267	1268	1269	1270	1271	1272	1273	1274	1275	1276	1277	1278	1279	1280	1281	1282	1283	1284	1285	1286	1287	1288	1289	1290	1291	1292	1293	1294	1295	1296	1297	1298	1299	1300	1301	1302	1303
8	0518	1304	1305	1306	1307	1308	1309	1310	1311	1312	1313	1314	1315	1316	1317	1318	1319	1320	1321	1322	1323	1324	1325	1326	1327	1328	1329	1330	1331	1332	1333	1334	1335	1336	1337	1338	1339	1340	1341	1342	1343
9	0540	1344	1345	1346	1347	1348	1349	1350	1351	1352	1353	1354	1355	1356	1357	1358	1359	1360	1361	1362	1363	1364	1365	1366	1367	1368	1369	1370	1371	1372	1373	1374	1375	1376	1377	1378	1379	1380	1381	1382	1383
10	0568	1384	1385	1386	1387	1388	1389	1390	1391	1392	1393	1394	1395	1396	1397	1398	1399	1400	1401	1402	1403	1404	1405	1406	1407	1408	1409	1410	1411	1412	1413	1414	1415	1416	1417	1418	1419	1420	1421	1422	1423
11	0590	1424	1425	1426	1427	1428	1429	1430	1431	1432	1433	1434	1435	1436	1437	1438	1439	1440	1441	1442	1443	1444	1445	1446	1447	1448	1449	1450	1451	1452	1453	1454	1455	1456	1457	1458	1459	1460	1461	1462	1463
12	05B8	1464	1465	1466	1467	1468	1469	1470	1471	1472	1473	1474	1475	1476	1477	1478	1479	1480	1481	1482	1483	1484	1485	1486	1487	1488	1489	1490	1491	1492	1493	1494	1495	1496	1497	1498	1499	1500	1501	1502	1503
13	05E0	1504	1505	1506	1507	1508	1509	1510	1511	1512	1513	1514	1515	1516	1517	1518	1519	1520	1521	1522	1523	1524	1525	1526	1527	1528	1529	1530	1531	1532	1533	1534	1535	1536	1537	1538	1539	1540	1541	1542	1543
14	0608	1544	1545	1546	1547	1548	1549	1550	1551	1552	1553	1554	1555	1556	1557	1558	1559	1560	1561	1562	1563	1564	1565	1566	1567	1568	1569	1570	1571	1572	1573	1574	1575	1576	1577	1578	1579	1580	1581	1582	1583
15	0630	1584	1585	1586	1587	1588	1589	1590	1591	1592	1593	1594	1595	1596	1597	1598	1599	1600	1601	1602	1603	1604	1605	1606	1607	1608	1609	1610	1611	1612	1613	1614	1615	1616	1617	1618	1619	1620	1621	1622	1623
16	0658	1624	1625	1626	1627	1628	1629	1630	1631	1632	1633	1634	1635	1636	1637	1638	1639	1640	1641	1642	1643	1644	1645	1646	1647	1648	1649	1650	1651	1652	1653	1654	1655	1656	1657	1658	1659	1660	1661	1662	1663
17	0680	1664	1665	1666	1667	1668	1669	1670	1671	1672	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682	1683	1684	1685	1686	1687	1688	1689	1690	1691	1692	1693	1694	1695	1696	1697	1698	1699	1700	1701	1702	1703
18	06A8	1704	1705	1706	1707	1708	1709	1710	1711	1712	1713	1714	1715	1716	1717	1718	1719	1720	1721	1722	1723	1724	1725	1726	1727	1728	1729	1730	1731	1732	1733	1734	1735	1736	1737	1738	1739	1740	1741	1742	1743
19	06D0	1744	1745	1746	1747	1748	1749	1750	1751	1752	1753	1754	1755	1756	1757	1758	1759	1760	1761	1762	1763	1764	1765	1766	1767	1768	1769	1770	1771	1772	1773	1774	1775	1776	1777	1778	1779	1780	1781	1782	1783
20	06F8	1784	1785	1786	1787	1788	1789	1790	1791	1792	1793	1794	1795	1796	1797	1798	1799	1800	1801	1802	1803	1804	1805	1806	1807	1808	1809	1810	1811	1812	1813	1814	1815	1816	1817	1818	1819	1820	1821	1822	1823
21	0720	1824	1825	1826	1827	1828	1829	1830	1831	1832	1833	1834	1835	1836	1837	1838	1839	1840	1841	1842	1843	1844	1845	1846	1847	1848	1849	1850	1851	1852	1853	1854	1855	1856	1857	1858	1859	1860	1861	1862	1863
22	0748	1864	1865	1866	1867	1868	1869	1870	1871	1872	1873	1874	1875	1876	1877	1878	1879	1880	1881	1882	1883	1884	1885	1886	1887	1888	1889	1890	1891	1892	1893	1894	1895	1896	1897	1898	1899	1900	1901	1902	1903
23	0770	1904	1905	1906	1907	1908	1909	1910	1911	1912	1913	1914	1915	1916	1917	1918	1919	1920	1921	1922	1923	1924	1925	1926	1927	1928	1929	1930	1931	1932	1933	1934	1935	1936	1937	1938	1939	1940	1941	1942	1943
24	0798	1944	1945	1946	1947	1948	1949	1950	1951	1952	1953	1954	1955	1956	1957	1958	1959	1960	1961	1962	1963	1964	1965	1966	1967	1968	1969	1970	1971	1972	1973	1974	1975	1976	1977	1978	1979	1980	1981	1982	1983
25	07C0	1984	1985	1986	1987	1988	1989	1990	1991	1992	1993	1994	1995	1996	1997	1998	1999	2000	2001	2002	2003	2004	2005	2006	2007	2008	2009	2010	2011	2012	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023

2023

Commodore 64 Colour Table Map

		55296																																				
1	D800	55296	55297	55298	55299	55300	55301	55302	55303	55304	55305	55306	55307	55308	55309	55310	55311	55312	55313	55314	55315	55316	55317	55318	55319	55320	55321	55322	55323	55324	55325	55326	55327	55328	55329	55330	55331	55332	55333	55334	55335
2	D828	55336	55337	55338	55339	55340	55341	55342	55343	55344	55345	55346	55347	55348	55349	55350	55351	55352	55353	55354	55355	55356	55357	55358	55359	55360	55361	55362	55363	55364	55365	55366	55367	55368	55369	55370	55371	55372	55373	55374	55375
3	D850	55376	55377	55378	55379	55380	55381	55382	55383	55384	55385	55386	55387	55388	55389	55390	55391	55392	55393	55394	55395	55396	55397	55398	55399	55400	55401	55402	55403	55404	55405	55406	55407	55408	55409	55410	55411	55412	55413	55414	55415
4	D878	55416	55417	55418	55419	55420	55421	55422	55423	55424	55425	55426	55427	55428	55429	55430	55431	55432	55433	55434	55435	55436	55437	55438	55439	55440	55441	55442	55443	55444	55445	55446	55447	55448	55449	55450	55451	55452	55453	55454	55455
5	D8A0	55456	55457	55458	55459	55460	55461	55462	55463	55464	55465	55466	55467	55468	55469	55470	55471	55472	55473	55474	55475	55476	55477	55478	55479	55480	55481	55482	55483	55484	55485	55486	55487	55488	55489	55490	55491	55492	55493	55494	55495
6	D8C8	55496	55497	55498	55499	55500	55501	55502	55503	55504	55505	55506	55507	55508	55509	55510	55511	55512	55513	55514	55515	55516	55517	55518	55519	55520	55521	55522	55523	55524	55525	55526	55527	55528	55529	55530	55531	55532	55533	55534	55535
7	D8F0	55536	55537	55538	55539	55540	55541	55542	55543	55544	55545	55546	55547	55548	55549	55550	55551	55552	55553	55554	55555	55556	55557	55558	55559	55560	55561	55562	55563	55564	55565	55566	55567	55568	55569	55570	55571	55572	55573	55574	55575
8	D918	55576	55577	55578	55579	55580	55581	55582	55583	55584	55585	55586	55587	55588	55589	55590	55591	55592	55593	55594	55595	55596	55597	55598	55599	55600	55601	55602	55603	55604	55605	55606	55607	55608	55609	55610	55611	55612	55613	55614	55615
9	D940	55616	55617	55618	55619	55620	55621	55622	55623	55624	55625	55626	55627	55628	55629	55630	55631	55632	55633	55634	55635	55636	55637	55638	55639	55640	55641	55642	55643	55644	55645	55646	55647	55648	55649	55650	55651	55652	55653	55654	55655
10	D968	55656	55657	55658	55659	55660	55661	55662	55663	55664	55665	55666	55667	55668	55669	55670	55671	55672	55673	55674	55675	55676	55677	55678	55679	55680	55681	55682	55683	55684	55685	55686	55687	55688	55689	55690	55691	55692	55693	55694	55695
11	D990	55696	55697	55698	55699	55700	55701	55702	55703	55704	55705	55706	55707	55708	55709	55710	55711	55712	55713	55714	55715	55716	55717	55718	55719	55720	55721	55722	55723	55724	55725	55726	55727	55728	55729	55730	55731	55732	55733	55734	55735
12	D9B8	55736	55737	55738	55739	55740	55741	55742	55743	55744	55745	55746	55747	55748	55749	55750	55751	55752	55753	55754	55755	55756	55757	55758	55759	55760	55761	55762	55763	55764	55765	55766	55767	55768	55769	55770	55771	55772	55773	55774	55775
13	D9E0	55776	55777	55778	55779	55780	55781	55782	55783	55784	55785	55786	55787	55788	55789	55790	55791	55792	55793	55794	55795	55796	55797	55798	55799	55800	55801	55802	55803	55804	55805	55806	55807	55808	55809	55810	55811	55812	55813	55814	55815
14	DA08	55816	55817	55818	55819	55820	55821	55822	55823	55824	55825	55826	55827	55828	55829	55830	55831	55832	55833	55834	55835	55836	55837	55838	55839	55840	55841	55842	55843	55844	55845	55846	55847	55848	55849	55850	55851	55852	55853	55854	55855
15	DA30	55856	55857	55858	55859	55860	55861	55862	55863	55864	55865	55866	55867	55868	55869	55870	55871	55872	55873	55874	55875	55876	55877	55878	55879	55880	55881	55882	55883	55884	55885	55886	55887	55888	55889	55890	55891	55892	55893	55894	55895
16	DA58	55896	55897	55898	55899	55900	55901	55902	55903	55904	55905	55906	55907	55908	55909	55910	55911	55912	55913	55914	55915	55916	55917	55918	55919	55920	55921	55922	55923	55924	55925	55926	55927	55928	55929	55930	55931	55932	55933	55934	55935
17	DA80	55936	55937	55938	55939	55940	55941	55942	55943	55944	55945	55946	55947	55948	55949	55950	55951	55952	55953	55954	55956	55956	55957	55958	55959	55960	55961	55962	55963	55964	55965	55966	55967	55968	55969	55970	55971	55972	55973	55974	55975
18	DAA8	55976	55977	55978	55979	55980	55981	55982	55983	55984	55985	55986	55987	55988	55989	55990	55991	55992	55993	55994	55995	55996	55997	55998	55999	56000	56001	56002	56003	56004	56005	56006	56007	56008	56009	56010	56011	56012	56013	56014	56015
19	DAD0	56016	56017	56018	56019	56020	56021	56022	56023	56024	56025	56026	56027	56028	56029	56030	56031	56032	56033	56034	56035	56036	56037	56038	56039	56040	56041	56042	56043	56044	56045	56046	56047	56048	56049	56050	56051	56052	56053	56054	56055
20	DAF8	56056	56057	56058	56059	56060	56061	56062	56063	56064	56065	56066	56067	56068	56069	56070	56071	56072	56073	56074	56075	56076	56077	56078	56079	56080	56081	56082	56083	56084	56085	56086	56087	56088	56089	56090	56091	56092	56093	56094	56095
21	DB20	56096	56097	56098	56099	56100	56101	56102	56103	56104	56105	56106	56107	56108	56109	56110	56111	56112	56113	56114	56115	56116	56117	56118	56119	56120	56121	56122	56123	56124	56125	56126	56127	56128	56129	56130	56131	56132	56133	56134	56135
22	DB48	56136	56137	56138	56139	56140	56141	56142	56143	56144	56145	56146	56147	56148	56149	56150	56151	56152	56153	56154	56155	56156	56157	56158	56159	56160	56161	56162	56163	56164	56165	56166	56167	56168	56169	56170	56171	56172	56173	56174	56175
23	DB70	56176	56177	56178	56179	56180	56181	56182	56183	56184	56185	56186	56187	56188	56189	56190	56191	56192	56193	56194	56195	56196	56197	56198	56199	56200	56201	56202	56203	56204	56205	56206	56207	56208	56209	56210	56211	56212	56213	56214	56215
24	DB98	56216	56217	56218	56219	56220	56221	56222	56223	56224	56225	56226	56227	56228	56229	56230	56231	56232	56233	56234	56235	56236	56237	56238	56239	56240	56241	56242	56243	56244	56245	56246	56247	56248	56249	56250	56251	56252	56253	56254	56255
25	DBC0	56256	56257	56258	56259	56260	56261	56262	56263	56264	56265	56266	56267	56268	56269	56270	56271	56272	56273	56274	56275	56276	56277	56278	56279	56280	56281	56282	56283	56284	56285	56286	56287	56288	56289	56290	56291	56292	56293	56294	56295

56295

VIC 20 Screen Memory

To move the screen: POKE 36869, (PEEK(36869) AND 15) OR X
POKE 36866, (PEEK(36866) AND 127) OR Y

X	Y	4*(PEEK(36866) AND 128) + 64*(PEEK(36869) AND 112) = Location	
		Decimal (½K blocks)	Hexadecimal
128	0	0	$0000
128	128	512	$0200
129	0	1024	0400
129	128	1536	0600
130	0	2048	0800
130	128	2560	0A00
131	0	3072	0C00
131	128	3584	0E00
132	0	4096	1000 (dflt w/exp)
132	128	4608	1200
133	0	5120	1400
133	128	5632	1600
134	0	6144	1800
134	128	6656	1A00
135	0	7168	1C00
135	128	7680	1E00 (default)
136	0	8192	2000
136	128	8704	2200
137	0	9216	2400
137	128	9728	2600
138	0	10240	2800
138	128	10752	2A00
139	0	11264	2C00
139	128	11776	2E00
140	0	12288	3000
140	128	12800	3200
141	0	13312	3400
141	128	13824	3600
142	0	14336	3800
142	128	14848	3A00
143	0	15360	3C00
143	128	15872	3E00

Commodore 64 Screen Memory

To move the screen: POKE 53272, (PEEK(53272)AND15) OR X

X	(3–PEEK(56576) AND 3) * 16384 + (X*64) = Location For Screen at Bank 0 (default):	
	Decimal	Hexadecimal
0	0	$0000
16	1024	0400 (default)
32	2048	0800
48	3072	0C00
64	4096	1000
80	5120	1400
96	6144	1800
112	7168	1C00
128	8192	2000
144	9216	2400
160	10240	2800
176	11264	2C00
192	12288	3000
208	13312	3400
224	14336	3800
240	15360	3C00

Commodore 64 VIC II Address

To move VIC II: POKE 53272, (PEEK(53272) AND 252) OR X ;X = 3–Bank#

Bank	X	VIC II Chip Address Range	
		Decimal (16K blocks)	Hexadecimal
0	3	0–16383	$0000–3FFF (default)
1	2	16384–32767	4000–7FFF
2	1	32768–49151	8000–BFFF
3	0	49152–65535	C000–FFFF

Note: Character ROM only available with VIC II in bank 0 or 2

VIC 20 Character Base

To move the character base: POKE 36869, (PEEK(36869) AND 240) OR X

X*	32768 + (PEEK(36869) AND 15) * 1024 = Location	
	Decimal (1K blocks)	Hexadecimal
0	32768–34815	$8000–87FF (dflt)
1	33792–35839	8400–8BFF
2	34816–36863	8800–8FFF
3	35840–37887	8C00–93FF
4	36864–38911	9000–97FF
5	37888–39935	9400–9BFF
6	38912–40959	9800–9FFF
7	39936–41983	9C00–A3FF
8	0–2047	0000–07FF
9	1024–3071	0400–0BFF
10	2048–4095	0800–0FFF
11	3072–5019	0C00–13FF
12	4096–6143	1000–17FF
13	5020–7167	1400–1BFF
14	6144–8191	1800–1FFF
15	7168–9216	1C00–23FF

* X = PEEK(36869) AND 15

Commodore 64 Character Base

To move the character base: POKE 53272, (PEEK(53272) AND 240) OR X

X*	(3–PEEK(56576) AND 3) * 16384 + (X*64) = Location For Screen at Bank 0 (default):	
	Decimal (2K blocks)	Hexadecimal
0	0–2047	$0000–07FF
2	2048–4095	0800–0FFF
4	4096–6143	1000–17FF *1
6	6144–8191	1800–1FFF *2
8	8192–10293	2000–27FF
10	10240–12287	2800–2FFF
12	12288–14335	3000–37FF
14	14336–16383	3800–3FFF

* – X = PEEK(53272) AND 14
*1 – Lower 2K of Character ROM (Bank 0 or 2 only) (default)
*2 – Upper 2K of Character ROM (Bank 0 or 2 only)

Character ROM Contents

Character ROM is the same in all machines, but only addressable in VIC 20/C64

2K Block	VIC 20		Commodore 64			Contents
	Default Address		Default Address		VIC II Image	
	Dec (½K blocks)	Hex	Dec (½K blocks)	Hex	Hex	
0	32768–33279	8000–81FF	53248–53759	D000–D1FF	1000–11FF	Upper case characters
	33280–33791	8200–83FF	53760–54271	D200–D3FF	1200–13FF	Graphics characters
	33792–34303	8400–85FF	54272–54783	D400–D5FF	1400–15FF	Reversed upper case characters
	34304–34815	8600–87FF	54784–55295	D600–D7FF	1600–17FF	Reversed graphics characters
1	34816–35327	8800–89FF	55296–55807	D800–D9FF	1800–19FF	Lower case characters
	35328–35839	8A00–8BFF	55808–56319	DA00–DBFF	1A00–1BFF	Upper case and graphics characters
	35840–36351	8C00–8DFF	56320–56831	DC00–DDFF	1C00–1DFF	Reversed lower case characters
	36352–36863	8E00–8FFF	56832–57343	DE00–DFFF	1E00–1FFF	Reversed upper case and graphics

80 Column Screen Map (8032, 8096, SuperPET)

| | | 32768 | | |

B Series Screen Map

		53248	55247
1	D000		
2	D050		
3	D0A0		
4	D0F0		
5	D140		
6	D190		
7	D1E0		
8	D230		
9	D280		
10	D2D0		
11	D320		
12	D370		
13	D3C0		
14	D410		
15	D460		
16	D4B0		
17	D500		
18	D550		
19	D5A0		
20	D5F0		
21	D640		
22	D690		
23	D6E0		
24	D730		
25	D780		

VIC 20 Screen & Border Colours

POKE 36879, X :	Border							
Screen	**BLK**	**WHT**	**RED**	**CYN**	**PUR**	**GRN**	**BLU**	**YEL**
BLK	8	9	10	11	12	13	14	15
WHT	24	25	26	27	28	29	30	31
RED	40	41	42	43	44	45	46	47
CYN	56	57	58	59	60	61	62	63
PUR	72	73	74	75	76	77	78	79
GRN	88	89	90	91	92	93	94	95
BLU	104	105	106	107	108	109	110	111
YEL	120	121	122	123	124	125	126	127
ORG	136	137	138	139	140	141	142	143
Lt. ORG	152	153	154	155	156	157	158	159
PNK	168	169	170	171	172	173	174	175
Lt. CYN	184	185	186	187	188	189	190	191
Lt. PUR	200	201	202	203	204	205	206	207
Lt. GRN	216	217	218	219	220	221	222	223
Lt. BLU	232	233	234	235	236	237	238	239
Lt. YEL	248	249	250	251	252	253	254	255

Mde: / Byts:	IMM 2	ZPg 2	Z,X 2	(I,X) 2	(I),Y 2	ABS 3	A,X 3	A,Y 3
ORA	09	05	15	01	11	0D	1D	19
AND	29	25	35	21	31	2D	3D	39
EOR	49	45	55	41	51	4D	5D	59
ADC	69	65	75	61	71	6D	7D	79
STA		85	95	81	91	8D	9D	99
LDA	A9	A5	B5	A1	B1	AD	BD	B9
CMP	C9	C5	D5	C1	D1	CD	DD	D9
SBC	E9	E5	F5	E1	F1	ED	FD	F9
Op Code ends in -1, -5, -9, or -D								

Mde: / Byts:	IMM 2	ZPg 2	Z,X 2	ABS 3	A,X 3
BIT		24		2C	
STY		84	94	8C	
LDY	A0	A4	B4	AC	BC
CPY	C0	C4		CC	
CPX	E0	E4		EC	
Op Code ends in -0, -4, or -C					

Jim Butterfield

Branches -0			
BPL	10	BMI	30
BVC	50	BVS	70
BCC	90	BCS	B0
BNE	D0	BEQ	F0

Jumps		
Mde:	ABS	(IND)
JSR	20	
JMP	4C	6C

Mde: / Byts:	IMM 2	ZPg 2	Z,X 2	Z,Y 2	ABS 3	A,X 3	A,Y 3
ASL		06	16		0E	1E	
ROL		26	36		2E	3E	
LSR		46	56		4E	5E	
ROR		66	76		6E	7E	
STX		86		96	8E		
LDX	A2	A6		B6	AE		BE
DEC		C6	D6		CE	DE	
INC		E6	F6		EE	FE	
Op Code ends in -2, -6, or -E							

Single Byte Op Codes (* Accumulator Mode)															
	0-	1-	2-	3-	4-	5-	6-	7-	8-	9-	A-	B-	C-	D-	E-	F-
-0	BRK				RTI		RTS									
-8	PHP	CLC	PLP	SEC	PHA	CLI	PLA	SEI	DEY	TYA	TAY	CLV	INY	CLD	INX	SED
-A	ASL*		ROL*		LSR*		ROR*		TXA	TXS	TAX	TSX	DEX		NOP	

Commodore 6845 Video Chip

POKE 59520, R#	POKE 59521, Value
R0	Horizontal total number of characters on line (Nht) including horizontal retrace. (true value = number + 1)
R1	Horizontal number of characters displayed (Nhd)
R2	Distance (in characters) from left to right margin of screen + 1
R3	Sync width. Lo nybble is vertical sync width (in lines) Hi nybble is horizontal sync (in characters).
R4	Number of display lines including retrace (Nvt).
R5	Vertical position of the edge of the screen.
R6	Number of display lines on screen (Nvd)
R7	Height of upper edge from bottom of screen (in lines displayed)
R8	Interlace and Skew:- Bit 0 1 = interlaced mode 0 = non interlaced mode Bit 1 if Bit 0 = 1 then interlace and video mode Bit 2 not used Bit 3 not used Bit 4 1 = scan from 32770 in memory Bit 5 1 = scan from 32772 in memory Bit 6 cursor (not implemented on the PET) Bit 7 cursor (not implemented on the PET)
R9	Number of lines between top of one display line and top of the next
R10	Cursor (not implemented on the PET)
R11	Cursor (not implemented on the PET)
R12	Control Register: Bit 0 add 256 to start address (512 for 8032) Bit 1 add 512 to start address (1024 for 8032) Bit 2 invert flyback Bit 3 invert video signal Bit 4 use top half of 4K character generator Bit 5 (not implemented on the PET) Bit 6 (not implemented on the PET) Bit 7 not used
R13	Value + 32768 is address of first character (multiply by 2 for 8032)
R14	Cursor location HI (not implemented on the PET)
R15	Cursor location LO (not implemented on the PET)
R16	Light pen position HI (read only)
R17	Light pen position LO (read only)

Hexadecimal Conversion Chart

Hex	-0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-A	-B	-C	-D	-E	-F	-00	-000
0-	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	0	0
1-	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	256	4096
2-	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	512	8192
3-	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	768	12288
4-	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	1024	16384
5-	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94	95	1280	20480
6-	96	97	98	99	100	101	102	103	104	105	106	107	108	109	110	111	1536	24576
7-	112	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	1792	28672
8-	128	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	2048	32768
9-	144	145	146	147	148	149	150	151	152	153	154	155	156	157	158	159	2304	36864
A-	160	161	162	163	164	165	166	167	168	169	170	171	172	173	174	175	2560	40960
B-	176	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	2816	45056
C-	192	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	3072	49152
D-	208	209	210	211	212	213	214	215	216	217	218	219	220	221	222	223	3328	53248
E-	224	225	226	227	228	229	230	231	232	233	234	235	236	237	238	239	3584	57344
F-	240	241	242	243	244	245	246	247	248	249	250	251	252	253	254	255	3840	61440

Bit Values

Bit	Dec	Hex
0	1	$0001
1	2	0002
2	4	0004
3	8	0008
4	16	0010
5	32	0020
6	64	0040
7	128	0080
8	256	0100
9	512	0200
10	1024	0400
11	2048	0800
12	4096	1000
13	8192	2000
14	16384	4000
15	32768	8000

Commodore 64 SID Note Values

The value under Hi is POKEd into the Hi byte of the frequency registers (54273, 54280, 54287). Likewise with Lo (54272, 54279, 54286)

Note	Octave 0			Octave 1			Octave 2			Octave 3		
	Oscillator Frequency			Oscillator Frequency			Oscillator Frequency			Oscillator Frequency		
	Decimal =	Hi (x256) + Lo		Decimal =	Hi (x256) + Lo		Decimal =	Hi (x256) + Lo		Decimal =	Hi (x256) + Lo	
C	268	1	12	536	2	24	1072	4	48	2145	8	97
C#	284	1	28	568	2	56	1136	4	112	2273	8	225
D	301	1	45	602	2	90	1204	4	180	2408	9	104
D#	318	1	62	637	2	125	1275	4	251	2551	9	247
E	337	1	81	675	2	163	1351	5	71	2703	10	143
F	358	1	102	716	2	204	1432	5	152	2864	11	48
F#	379	1	123	758	2	246	1517	5	237	3034	11	218
G	401	1	145	803	3	35	1607	6	71	3215	12	143
G#	425	1	169	851	3	83	1703	6	167	3406	13	78
A	451	1	195	902	3	134	1804	7	12	3608	14	24
A#	477	1	221	955	3	187	1911	7	119	3823	14	239
B	506	1	250	1012	3	244	2025	7	233	4050	15	210

Note	Octave 4			Octave 5			Octave 6			Octave 7		
	Oscillator Frequency			Oscillator Frequency			Oscillator Frequency			Oscillator Frequency		
	Decimal =	Hi (x256) + Lo		Decimal =	Hi (x256) + Lo		Decimal =	Hi (x256) + Lo		Decimal =	Hi (x256) + Lo	
C	4291	16	195	8583	33	135	17167	67	15	34334	134	30
C#	4547	17	195	9094	35	134	18188	71	12	36376	142	24
D	4817	18	209	9634	37	162	19269	75	69	38539	150	139
D#	5103	19	239	10207	39	223	20415	79	191	40830	159	126
E	5407	21	31	10814	42	62	21629	84	125	43258	168	250
F	5728	22	96	11457	44	193	22915	89	131	45830	179	6
F#	6069	23	181	12139	47	107	24278	94	214	48556	189	172
G	6430	25	30	12860	50	60	25721	100	121	51443	200	243
G#	6812	26	156	13625	53	57	27251	106	115	54502	212	230
A	7217	28	49	14435	56	99	28871	112	199	57743	225	143
A#	7647	29	223	15294	59	190	30588	119	124	61176	238	248
B	8101	31	165	16203	63	75	32407	126	151	64814	253	46

CB2 Note Values

Values are for simple CB2 sound.

PET/CBM : POKE 59467,16 : POKE 59466, (Oct) : POKE 59464, X
VIC 20 : POKE 37147,16 : POKE 37146, (Oct) : POKE 37144, X
C64 : POKE 56587,16 : POKE 56586, (Oct) : POKE 56584, X

Note	Oct = 15		Oct = 51		Oct = 85	
	Octave 0	Octave 1	Octave 2	Octave 3	Octave 4	Octave 5
B	251[b]	125	251	125	251	125
C	238	118	238	118	238	118
C#	224	110	224	110	224	110
D	210	104	210	104	210	104
D#	199	99	199	99	199	99
E	188	93	188	93	188	93
F	177	88	177	88	177	88
F#	168	83	168	83	168	83
G	158	78	158	78	158	78
G#	149	74	149	74	149	74
A	140	69	140	69	140	69
A#	133	65	133	65	133	65

Based on formula: Note n = (Note n–1) / 2 ↑ (1/12)
Reset Port with first POKE (above) ,0

Commodore 64 SID Envelope Rates

Master Volume (MV) = Lo nybble of 54296. MV and A/D/S/R Registers (R1 & R2) are write only.
Voice1: 54277/8 • Voice2: 54284/5 • Voice3: 54291/2. See Memory Map.

Value		POKE R1, (Hi + Lo)		POKE R2, (Hi + Lo)	
		Hi nybble	Lo nybble	Hi nybble	Lo nybble
Hi nybble	Lo nybble	Attack Rate 0 to peak	Decay Rate peak to SL	Sustain Level Val/15th's of MV	Release rate SL to 0
0	0	2 ms	6 ms	0/15MV	6 ms
16	1	8 ms	24 ms	1/15MV	24 ms
32	2	16 ms	48 ms	2/15MV	48 ms
48	3	24 ms	72 ms	3/15MV	72 ms
64	4	38 ms	114 ms	4/15MV	114 ms
80	5	56 ms	168 ms	5/15MV	168 ms
96	6	68 ms	204 ms	6/15MV	204 ms
112	7	80 ms	240 ms	7/15MV	240 ms
128	8	100 ms	300 ms	8/15MV	300 ms
144	9	250 ms	750 ms	9/15MV	750 ms
160	10	500 ms	1.5 s	10/15MV	1.5 s
176	11	800 ms	2.4 s	11/15MV	2.4 s
192	12	1.0 s	3.0 s	12/15MV	3.0 s
208	13	3.0 s	9.0 s	13/15MV	9.0 s
224	14	5.0 s	15.0 s	14/15MV	15.0 s
240	15	8.0 s	24.0 s	= MV	24.0 s

VIC 20 Note Values

Where two values are shown, it is necessary to alternate between them to get the true note.
Voice frequency registers are 36874/5/6. Noise reg is 36877. Volume is Lo nybble of 36878. See Memory Map

Note	Octave 0		Octave 1		Octave 2		Octave 3	
	Value	Mod. Val.	Value	Mod. Val.	Value	Mod. Val.	Value	Mod. Val.
C	131		192	195	224		239	240
C#	140		197		226		240	241
D	145		200		227	228		
D#	151		203		229			
E	158		206	207	231			
F	161	162	208	209	232			
F#	166	167	211	212	233			
G	173	174	214		234	235		
G#	178		216		238	236		
A	181	182	218	219	237			
A#	185	186	220	221	237	238		
B	189	190	222	223	239			

SuperChart: VIC 20 / Commodore 64

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
0	00		@	end-line	BRK	0
1	01		A		ORA(I,X)	1
2	02		B			2
3	03	stop	C			3
4	04		D			4
5	05	white	E		ORA Z	5
6	06		F		ASL Z	6
7	07		G			7
8	08	lock	H		PHP	8
9	09	unlock	I		ORA #	9
10	0A		J		ASL A	10
11	0B		K			11
12	0C		L			12
13	0D	car ret	M		ORA	13
14	0E	text	N		ASL	14
15	0F		O			15
16	10		P		BPL	16
17	11	cur down	Q		ORA(I),Y	17
18	12	reverse	R			18
19	13	cur home	S			19
20	14	delete	T			20
21	15		U		ORA Z,X	21
22	16		V		ASL Z,X	22
23	17		W			23
24	18		X		CLC	24
25	19		Y		ORA Y	25
26	1A		Z			26
27	1B		[			27
28	1C	red	\			28
29	1D	cur right	]		ORA X	29
30	1E	green	↑		ASL X	30
31	1F	blue	←			31
32	20	space	space	space	JSR	32
33	21	!	!	!	AND(I,X)	33
34	22	"	"	"		34
35	23	#	#	#		35
36	24	$	$	$	BIT Z	36
37	25	%	%	%	AND Z	37
38	26	&	&	&	ROL Z	38
39	27	'	'	'		39
40	28	(	(	(	PLP	40
41	29	)	)	)	AND #	41
42	2A	*	*	*	ROL A	42
43	2B	+	+	+		43
44	2C	,	,	,	BIT	44
45	2D	-	-	-	AND	45
46	2E	.	.	.	ROL	46
47	2F	/	/	/		47
48	30	0	0	0	BMI	48
49	31	1	1	1	AND(I),Y	49
50	32	2	2	2		50
51	33	3	3	3		51
52	34	4	4	4		52
53	35	5	5	5	AND Z,X	53
54	36	6	6	6	ROL Z,X	54
55	37	7	7	7		55
56	38	8	8	8	SEC	56
57	39	9	9	9	AND Y	57
58	3A	:	:	:		58
59	3B	;	;	;		59
60	3C	<	<	<		60
61	3D	=	=	=	AND X	61
62	3E	>	>	>	ROL X	62
63	3F	?	?	?		63

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
64	40	@	─	@	RTI	64
65	41	A	♠,a	A	EOR(I,X)	65
66	42	B	▏,b	B		66
67	43	C	▔,c	C		67
68	44	D	🮂,d	D		68
69	45	E	▕,e	E	EOR Z	69
70	46	F	▁,f	F	LSR Z	70
71	47	G	▏,g	G		71
72	48	H	▒,h	H	PHA	72
73	49	I	🮈,i	I	EOR #	73
74	4A	J	🮉,j	J	LSR A	74
75	4B	K	🮊,k	K		75
76	4C	L	▗,l	L	JMP	76
77	4D	M	▚,m	M	EOR	77
78	4E	N	▞,n	N	LSR	78
79	4F	O	▖,o	O		79
80	50	P	▘,p	P	BVC	80
81	51	Q	▝,q	Q	EOR(I),Y	81
82	52	R	▂,r	R		82
83	53	S	▌,s	S		83
84	54	T	▀,t	T		84
85	55	U	🮇,u	U	EOR Z,X	85
86	56	V	▐,v	V	LSR Z,X	86
87	57	W	▄,w	W		87
88	58	X	▙,x	X	CLI	88
89	59	Y	▟,y	Y	EOR Y	89
90	5A	Z	◆,z	Z		90
91	5B	[	┼	[		91
92	5C	£	▏	£		92
93	5D	]	▕	]	EOR X	93
94	5E	↑	π,▚	↑	LSR X	94
95	5F	←	◥,▨	←		95
96	60		▔		RTS	96
97	61		▏		ADC(I,X)	97
98	62		▂			98
99	63		▔			99
100	64		▁			100
101	65		▔		ADC Z	101
102	66		▉		ROR Z	102
103	67		▕			103
104	68		▃		PLA	104
105	69		◩,▨		ADC #	105
106	6A		▐		ROR A	106
107	6B		▐			107
108	6C		▄		JMP(I)	108
109	6D		▕		ADC	109
110	6E		▕		ROR	110
111	6F		▁			111
112	70		▔		BVS	112
113	71		▔		ADC(I),Y	113
114	72		▔			114
115	73		▔			115
116	74		▏			116
117	75		▏		ADC Z,X	117
118	76		▐		ROR Z,X	118
119	77		▔			119
120	78		▔		SEI	120
121	79		▃		ADC Y	121
122	7A		▁,▩			122
123	7B		▕			123
124	7C		▔			124
125	7D		▔		ADC X	125
126	7E		▔		ROR X	126
127	7F		▉			127

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
128	80		@	END		128
129	81	orange	A	FOR	STA(I,X)	129
130	82		B	NEXT		130
131	83	load & run	C	DATA		131
132	84		D	INPUT#	STY Z	132
133	85	f1	E	INPUT	STA Z	133
134	86	f2	F	DIM	STX Z	134
135	87	f3	G	READ		135
136	88	f4	H	LET	DEY	136
137	89	f5	I	GOTO		137
138	8A	f6	J	RUN	TXA	138
139	8B	f7	K	IF		139
140	8C	f8	L	RESTORE	STY	140
141	8D	car ret	M	GOSUB	STA	141
142	8E	graphics	N	RETURN	STX	142
143	8F		O	REM		143
144	90	black	P	STOP	BCC	144
145	91	cur up	Q	ON	STA(I),Y	145
146	92	rvs off	R	WAIT		146
147	93	clear	S	LOAD		147
148	94	insert	T	SAVE	STY Z,X	148
149	95	brown	U	VERIFY	STA Z,X	149
150	96	lt. red	V	DEF	STX Z,Y	150
151	97	dk. grey	W	POKE		151
152	98	md. grey	X	PRINT#	TYA	152
153	99	lt. green	Y	PRINT	STA Y	153
154	9A	lt. blue	Z	CONT	TXS	154
155	9B	lt. grey	[	LIST		155
156	9C	magenta	£	CLR		156
157	9D	cur left	]	CMD	STA X	157
158	9E	yellow	↑	SYS		158
159	9F	cyan	←	OPEN		159
160	A0	□		CLOSE	LDY #	160
161	A1	◧	!	GET	LDA(I,X)	161
162	A2	▬	"	NEW	LDX #	162
163	A3	□	#	TAB(		163
164	A4	□	$	TO	LDY Z	164
165	A5	□	%	FN	LDA Z	165
166	A6	■	&	SPC(	LDX Z	166
167	A7	□	'	THEN		167
168	A8	▬	(	NOT	TAY	168
169	A9	◩,▨	)	STEP	LDA #	169
170	AA	◨	*	+	TAX	170
171	AB	⊞	+	–		171
172	AC	▪	,	*	LDY	172
173	AD	◳	-	/	LDA	173
174	AE	◱	.	↑	LDX	174
175	AF	▭	/	AND		175
176	B0	◲	0	OR	BCS	176
177	B1	⊟	1	>	LDA(I),Y	177
178	B2	⊞	2	=		178
179	B3	⊞	3	<		179
180	B4	◧	4	SGN	LDY Z,X	180
181	B5	◧	5	INT	LDA Z,X	181
182	B6	◨	6	ABS	LDX Z,Y	182
183	B7	▀	7	USR		183
184	B8	▀	8	FRE	CLV	184
185	B9	▬	9	POS	LDA Y	185
186	BA	□,◪	:	SQR	TSX	186
187	BB	◧	;	RND		187
188	BC	◪	<	LOG	LDY X	188
189	BD	◰	=	EXP	LDA X	189
190	BE	◪	>	COS	LDX Y	190
191	BF	▪	?	SIN		191

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
192	C0	⊟		TAN	CPY #	192
193	C1	●,a		ATN	CMP(I),X	193
194	C2	◧,b		PEEK		194
195	C3	⊟,c		LEN		195
196	C4	⊟,d		STR$	CPY Z	196
197	C5	□,e		VAL	CMP Z	197
198	C6	▭,f		ASC	DEC Z	198
199	C7	◧,g		CHR$		199
200	C8	◨,h		LEFT$	INY	200
201	C9	◲,i		RIGHT$	CMP #	201
202	CA	◳,j		MID$	DEX	202
203	CB	◰,k		GO		203
204	CC	□,l			CPY	204
205	CD	▨,m			CMP	205
206	CE	▨,n			DEC	206
207	CF	□,o				207
208	D0	❑,p			BNE	208
209	D1	●,q			CMP(I),Y	209
210	D2	□,r				210
211	D3	♥,s				211
212	D4	□,t				212
213	D5	◲,u			CMP Z,X	213
214	D6	☒ v			DEC Z,X	214
215	D7	❑,w				215
216	D8	♣,x			CLD	216
217	D9	❑,y			CMP Y	217
218	DA	◆,z				218
219	DB	⊞				219
220	DC	◧				220
221	DD	◫			CMP X	221
222	DE	π,▩			DEC X	222
223	DF	◥,▨				223
224	E0		■		CPX #	224
225	E1		◨		SBC(I),X	225
226	E2		◧			226
227	E3		■			227
228	E4		■		CPX Z	228
229	E5		■		SBC Z	229
230	E6		■		INC Z	230
231	E7		■			231
232	E8		■		INX	232
233	E9		◪,▨		SBC #	233
234	EA		■		NOP	234
235	EB		■			235
236	EC		■		CPX	236
237	ED		■		SBC	237
238	EE		■		INC	238
239	EF		■			239
240	F0		■		BEQ	240
241	F1		■		SBC(I),Y	241
242	F2		■			242
243	F3		■			243
244	F4		■			244
245	F5		■		SBC Z,X	245
246	F6		■		INC Z,X	246
247	F7		■			247
248	F8		■		SED	248
249	F9		■		SBC Y	249
250	FA		■,◪			250
251	FB		■			251
252	FC		■			252
253	FD		■		SBC X	253
254	FE		■		INC X	254
255	FF		■ π			255

Reverse of ASCII

SuperChart: BASIC 2.0 / 4.0

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
0	00		@	end-line	BRK	0
1	01		A		ORA(I,X)	1
2	02		B			2
3	03	stop	C			3
4	04		D			4
5	05		E		ORA Z	5
6	06		F		ASL Z	6
7	07	bell	G			7
8	08		H		PHP	8
9	09	tab	I		ORA #	9
10	0A		J		ASL A	10
11	0B		K			11
12	0C		L			12
13	0D	car ret	M		ORA	13
14	0E	text	N		ASL	14
15	0F	top left	O			15
16	10		P		BPL	16
17	11	cur down	Q		ORA(I),Y	17
18	12	reverse	R			18
19	13	cur home	S			19
20	14	delete	T			20
21	15	del line	U		ORA Z,X	21
22	16	ers start	V		ASL Z,X	22
23	17		W			23
24	18		X		CLC	24
25	19	scroll dn	Y		ORA Y	25
26	1A		Z			26
27	1B	escape	[			27
28	1C		\			28
29	1D	cur right	]		ORA X	29
30	1E		↑		ASL X	30
31	1F		←			31
32	20	space	space	space	JSR	32
33	21	!	!	!	AND(I,X)	33
34	22	"	"	"		34
35	23	#	#	#		35
36	24	$	$	$	BIT Z	36
37	25	%	%	%	AND Z	37
38	26	&	&	&	ROL Z	38
39	27	'	'	'		39
40	28	(	(	(	PLP	40
41	29	)	)	)	AND #	41
42	2A	*	*	*	ROL A	42
43	2B	+	+	+		43
44	2C	,	,	,	BIT	44
45	2D	-	-	-	AND	45
46	2E	.	.	.	ROL	46
47	2F	/	/	/		47
48	30	0	0	0	BMI	48
49	31	1	1	1	AND(I),Y	49
50	32	2	2	2		50
51	33	3	3	3		51
52	34	4	4	4		52
53	35	5	5	5	AND Z,X	53
54	36	6	6	6	ROL Z,X	54
55	37	7	7	7		55
56	38	8	8	8	SEC	56
57	39	9	9	9	AND Y	57
58	3A	:	:	:		58
59	3B	;	;	;		59
60	3C	<	<	<		60
61	3D	=	=	=	AND X	61
62	3E	>	>	>	ROL X	62
63	3F	?	?	?		63

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
64	40	@	⊟	@	RTI	64
65	41	A	■,a	A	EOR(I,X)	65
66	42	B	⊞,b	B		66
67	43	C	⊟,c	C		67
68	44	D	⊟,d	D		68
69	45	E	□,e	E	EOR Z	69
70	46	F	⊟,f	F	LSR Z	70
71	47	G	⊞,g	G		71
72	48	H	◨,h	H	PHA	72
73	49	I	◱,i	I	EOR #	73
74	4A	J	◳,j	J	LSR A	74
75	4B	K	◰,k	K		75
76	4C	L	□,l	L	JMP	76
77	4D	M	▧,m	M	EOR	77
78	4E	N	◰,n	N	LSR	78
79	4F	O	□,o	O		79
80	50	P	□,p	P	BVC	80
81	51	Q	●,q	Q	EOR(I),Y	81
82	52	R	□,r	R		82
83	53	S	▼,s	S		83
84	54	T	□,t	T		84
85	55	U	◲,u	U	EOR Z,X	85
86	56	V	☒,v	V	LSR Z,X	86
87	57	W	□,w	W	•	87
88	58	X	♣,x	X	CLI	88
89	59	Y	□,y	Y	EOR Y	89
90	5A	Z	◆,z	Z		90
91	5B	[	⊞	[		91
92	5C	\	◧	\		92
93	5D	]	◫	]	EOR X	93
94	5E	↑	▣,▩	↑	LSR X	94
95	5F	←	◥,▨	←		95
96	60		□		RTS	96
97	61		◫		ADC(I,X)	97
98	62		⊟			98
99	63		□			99
100	64		□			100
101	65		□		ADC Z	101
102	66		■		ROR Z	102
103	67		□			103
104	68		⬓		PLA	104
105	69		◩,▨		ADC #	105
106	6A		◨		ROR A	106
107	6B		⊞			107
108	6C		◪		JMP(I)	108
109	6D		◳		ADC	109
110	6E		◱		ROR	110
111	6F		□			111
112	70		◲		BVS	112
113	71		⊞		ADC(I),Y	113
114	72		⊞			114
115	73		⊞			115
116	74		◧			116
117	75		◧		ADC Z,X	117
118	76		◨		ROR Z,X	118
119	77		◻			119
120	78		⬒		SEI	120
121	79		⬓		ADC Y	121
122	7A		□,☑			122
123	7B		◾			123
124	7C		◔			124
125	7D		◰		ADC X	125
126	7E		◾		ROR X	126
127	7F		▪			127

Jim Butterfield

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
128	80		@	END		128
129	81		A	FOR	STA(I,X)	129
130	82		B	NEXT		130
131	83	load & run	C	DATA		131
132	84		D	INPUT#	STY Z	132
133	85		E	INPUT	STA Z	133
134	86		F	DIM	STX Z	134
135	87	bell	G	READ		135
136	88		H	LET	DEY	136
137	89	set/clr tab	I	GOTO		137
138	8A		J	RUN	TXA	138
139	8B		K	IF		139
140	8C		L	RESTORE	STY	140
141	8D	car ret	M	GOSUB	STA	141
142	8E	graphics	N	RETURN	STX	142
143	8F	bot right	O	REM		143
144	90		P	STOP	BCC	144
145	91	cur up	Q	ON	STA(I),Y	145
146	92	rvs off	R	WAIT		146
147	93	clear	S	LOAD		147
148	94	insert	T	SAVE	STY Z,X	148
149	95	ins line	U	VERIFY	STA Z,X	149
150	96	ers end	V	DEF	STX Z,Y	150
151	97		W	POKE		151
152	98		X	PRINT#	TYA	152
153	99	scroll up	Y	PRINT	STA Y	153
154	9A		Z	CONT	TXS	154
155	9B	escape	[	LIST		155
156	9C		\	CLR		156
157	9D	cur left	]	CMD	STA X	157
158	9E		↑	SYS		158
159	9F		←	OPEN		159
160	A0			CLOSE	LDY #	160
161	A1		!	GET	LDA(I,X)	161
162	A2		"	NEW	LDX #	162
163	A3		#	TAB(		163
164	A4		$	TO	LDY Z	164
165	A5		%	FN	LDA Z	165
166	A6		&	SPC(	LDX Z	166
167	A7		'	THEN		167
168	A8		(	NOT	TAY	168
169	A9		)	STEP	LDA #	169
170	AA		*	+	TAX	170
171	AB		+	-		171
172	AC		,	*	LDY	172
173	AD		-	/	LDA	173
174	AE		.	↑	LDX	174
175	AF		/	AND		175
176	B0		0	OR	BCS	176
177	B1		1	>	LDA(I),Y	177
178	B2		2	=		178
179	B3		3	<		179
180	B4		4	SGN	LDY Z,X	180
181	B5		5	INT	LDA Z,X	181
182	B6		6	ABS	LDX Z,Y	182
183	B7		7	USR		183
184	B8		8	FRE	CLV	184
185	B9		9	POS	LDA Y	185
186	BA		:	SQR	TSX	186
187	BB		;	RND		187
188	BC		<	LOG	LDY X	188
189	BD		=	EXP	LDA X	189
190	BE		>	COS	LDX Y	190
191	BF		?	SIN		191

DECIMAL	HEX	ASCII	SCREEN	BASIC	6502	DECIMAL
192	C0			TAN	CPY #	192
193	C1	,a		ATN	CMP(I),X	193
194	C2	,b		PEEK		194
195	C3	,c		LEN		195
196	C4	,d		STR$	CPY Z	196
197	C5	,e		VAL	CMP Z	197
198	C6	,f		ASC	DEC Z	198
199	C7	,g		CHR$		199
200	C8	,h		LEFT$	INY	200
201	C9	,i		RIGHT$	CMP #	201
202	CA	,j		MID$	DEX	202
203	CB	,k		GO		203
204	CC	,l		CONCAT	CPY	204
205	CD	,m		DOPEN	CMP	205
206	CE	,n		DCLOSE	DEC	206
207	CF	,o		RECORD		207
208	D0	,p		HEADER	BNE	208
209	D1	,q		COLLECT	CMP(I),Y	209
210	D2	,r		BACKUP		210
211	D3	,s		COPY		211
212	D4	,t		APPEND		212
213	D5	,u		DSAVE	CMP Z,X	213
214	D6	,v		DLOAD	DEC Z,X	214
215	D7	,w		CATALOG		215
216	D8	,x		RENAME	CLD	216
217	D9	,y		SCRATCH	CMP Y	217
218	DA	,z		DIRECTORY		218
219	DB					219
220	DC					220
221	DD				CMP X	221
222	DE	,			DEC X	222
223	DF	,				223
224	E0				CPX #	224
225	E1				SBC(I),X	225
226	E2					226
227	E3					227
228	E4				CPX Z	228
229	E5				SBC Z	229
230	E6				INC Z	230
231	E7					231
232	E8				INX	232
233	E9				SBC #	233
234	EA				NOP	234
235	EB					235
236	EC				CPX	236
237	ED				SBC	237
238	EE				INC	238
239	EF					239
240	F0				BEQ	240
241	F1				SBC(I),Y	241
242	F2					242
243	F3					243
244	F4					244
245	F5				SBC Z,X	245
246	F6				INC Z,X	246
247	F7					247
248	F8				SED	248
249	F9				SBC Y	249
250	FA	,				250
251	FB					251
252	FC					252
253	FD				SBC X	253
254	FE				INC X	254
255	FF	π		π		255

Reverse of ASCII

BASIC 2.0 / BASIC 4.0 Memory Map

Jim Butterfield, Toronto Ont.

Reference to DOS, MLM, 80-Column, or those marked with an * are for BASIC 4.0 only. There are some differences between usage for the 40- and 80-column machines. 32K BASIC 4.0 Zero Page contents for 4000 and 8000 series machines after power up by Richard Evers. BASIC 2.0 Zero Page contents are mostly identical except for vectors.

Location				Contents				Description
Hex		Dec		4000 Hex	4000 Dec	8000 Hex	8000 Dec	
00 –02	00	0–2	0	4C	76	4C	76	USR jump instruction
	01		1	73	115	73	115	;JMP $C373
	02		2	C3	195	C3	195	
03	03	3	3	22	34	22	34	Search character
04	04	4	4	00	0	00	0	Scan-between-quotes flag
05	05	5	5	5B	91	5B	91	Input buffer pointer
06	06	6	6	00	0	FF	255	Default DIM flag
07	07	7	7	00	0	00	0	Type: $FF = string, $00 = numeric
08	08	8	8	00	0	00	0	Type: $80 = integer, 00 = floating pt
09	09	9	9	04	4	04	4	Flag: DATA scan; LIST quote; memory
0A	0A	10	10	00	0	00	0	Subscript flag; FNx flag
0B	0B	11	11	00	0	00	0	0 = INPUT; $40 = GET; $98 = READ
0C	0C	12	12	00	0	FF	255	ATN sign/comparison evaluation flag
0D–0F	0D	13–15	13	00	0	00	0	Disk status DS$ descriptor
	0F		14	FF	255	FF	255	
	0F		15	00	0	00	0	
10	10	16	16	00	0	00	0	Current I/O prompt flag
11 –12	11	17–18	17	72	114	72	114	Integer value (for SYS, GOTO etc.)
	12		18	D4	212	D4	212	
13 –15	13	19–21	19	16	22	16	22	Pointers for descriptor stack
	14		20	13	19	13	19	
	15		21	00	0	00	0	
16 –1E	16	22–30	22	08	8	08	8	Descriptor stack (temporary strings)
	17		23	12	18	12	18	
	18		24	B3	179	B3	179	
	19		25	00	0	00	0	
	1A		26	FF	255	FF	255	
	1B		27	00	0	00	0	
	1C		28	FF	255	FF	255	
	1D		29	00	0	00	0	
	1E		30	FF	255	FF	255	
1F –22	1F	31–34	31	40	64	40	64	Utility pointer area
	20		32	B2	178	B2	178	
	21		33	E9	233	E9	233	
	22		34	CE	206	CE	206	
23 –27	23	35–39	35	48	72	00	0	Product area for multiplication
	24		36	00	0	FF	255	
	25		37	00	0	00	0	
	26		38	00	0	FF	255	
	27		39	00	0	00	0	
28 –29	28	40–41	40	01	1	01	1	**Pointer:** Start of BASIC
	29		41	04	4	04	4	;0401
2A –2B	2A	42–43	42	03	3	03	3	**Pointer:** Start of Variables
	2B		43	04	4	04	4	;0403
2C –2D	2C	44–45	44	03	3	03	3	**Pointer:** Start of Arrays
	2D		45	04	4	04	4	;0403
2E –2F	2E	46–47	46	03	3	03	3	**Pointer:** End of Arrays
	2F		47	04	4	04	4	;0403
30 –31	30	48–49	48	00	0	00	00	**Pointer:** String Storage (moving down)
	31		49	80	128	80	128	;8000
32 –33	32	50–51	50	FE	254	FF	255	**Pointer:** String Utility
	33		51	7F	127	00	0	
34 –35	34	52–53	52	00	0	00	0	**Pointer:** Limit of Memory
	35		53	80	128	80	128	;8000
36 –37	36	54–55	54	14	20	FF	255	Current BASIC line number
	37		55	FF	255	FF	255	
38 –39	38	56–57	56	00	0	FF	255	Previous BASIC line number
	39		57	80	128	00	0	
3A –3B	3A	58–59	58	01	1	FF	255	**Pointer:** BASIC statement for CONT
	3B		59	00	0	00	0	
3C –3D	3C	60–61	60	00	0	FF	255	Current DATA line number
	3D		61	50	80	00	0	
3E –3F	3E	62–63	62	00	0	00	0	Current data address
	3F		63	04	4	04	4	
40 –41	40	64–65	64	00	0	FF	255	Input vector
	41		65	00	0	00	0	
42 –43	42	66–67	66	04	4	FF	255	Current variable name
	43		67	00	0	00	0	
44 –45	44	68–69	68	24	36	24	36	Current variable address
	45		69	04	4	00	0	
46 –47	46	70–71	70	82	130	FF	255	Variable pointer for FOR/NEXT
	47		71	04	4	00	0	
48 –49	48	72–73	72	FF	255	FF	255	Y-save; op-save; BASIC pointer save
	49		73	00	0	00	0	
4A	4A	74	74	00	0	00	0	Comparison symbol accumulator
4B –50	4B	75–80	75	00	0	00	0	Miscellaneous work area, pointers, etc.

Location				Contents				Description
Hex		Dec		4000 Hex	4000 Dec	8000 Hex	8000 Dec	
	4C		76	FF	255	FF	255	
	4D		77	16	22	00	0	
	4E		78	00	0	FF	255	
	4F		79	00	0	00	0	
	50		80	03	3	03	3	
51 –53	51	81–83	81	4C	76	4C	76	Jump vector for functions
	52		82	43	67	FF	255	
	53		83	00	0	00	0	
54 –5D	54	84–93	84	FF	255	FF	255	Miscellaneous numeric work area
	55		85	87	135	00	0	
	56		86	04	4	FF	255	
	57		87	80	128	00	0	
	58		88	03	3	FF	255	
	59		89	00	0	00	0	
	5A		90	00	0	00	0	
	5B		91	00	0	00	0	
	5C		92	00	0	00	0	
	5D		93	00	0	00	0	
5E	5E	94	94	90	144	94	144	Accum#1: Exponent
5F –62	5F	95–98	95	00	0	00	0	Accum#1: Mantissa
	60		96	00	0	00	0	
	61		97	D4	212	D4	212	
	62		98	72	114	72	114	
63	63	99	99	00	0	00	0	Accum#1: Sign
64	64	100	100	00	0	00	0	Series evaluation constant pointer
65	65	101	101	00	0	00	0	Accum#1 hi-order (overflow)
66 –6B	66	102–107	102	90	144	90	144	Accum#2: Exponent
	67		103	D4	212	D4	212	Accum#2: Mantissa
	68		104	6C	108	6C	108	
	69		105	00	0	00	0	
	6A		106	00	0	00	0	
	6B		107	00	0	00	0	Accum#2: Sign
6C	6C	108	108	00	0	00	0	Sign comparison, Acc#1 vs #2
6D	6D	109	109	00	0	00	0	Accum#1 lo-order (rounding)
6E –6F	6E	110–111	110	0A	10	0A	10	Cassette buff len/series pointer
	6F		111	B3	179	B3	179	
70 –87	70	112–135	112	E6	230	E6	230	CHRGET subroutine; get BASIC char
	71		113	77	119	77	119	;INC $77
	72		114	D0	208	D0	208	;BNE $0076
	73		115	02	2	02	2	
	74		116	E6	230	E6	230	;INC $78
	75		117	78	120	78	120	
	76		118	AD	173	AD	173	;LDA $0202
	77		119	02	2	02	2	
	78		120	02	2	02	2	
	79		121	C9	201	C9	201	;CMP #$3A
	7A		122	3A	58	3A	58	
	7B		123	B0	176	B0	176	;BCS $0087
	7C		124	0A	10	0A	10	
	7D		125	C9	201	C9	201	;CMP #$20
	7E		126	20	32	20	32	
	7F		127	F0	240	F0	240	;BEQ $0070
	80		128	EF	239	EF	239	
	81		129	38	56	38	56	;SEC
	82		130	E9	233	E9	233	;SBC #$30
	83		131	30	48	30	48	
	84		132	38	56	38	56	;SEC
	85		133	E9	233	E9	233	;SBC #$D0
	86		134	D0	208	D0	208	
	87		135	60	96	60	96	;RTS
77 –78	77	119–120	119	02	2	02	2	BASIC pointer (within subroutine)
	78		120	02	2	02	2	
88 –8C	88	136–140	136	80	128	80	128	Random number seed
	89		137	4F	79	4F	79	
	8A		138	C7	199	C7	199	
	8B		139	52	82	52	82	
	8C		140	F4	244	FF	255	
8D –8F	8D	141–143	141	00	0	00	0	Jiffy clock for TI and TI$
	8E		142	15	21	08	8	
	8F		143	89	137	1F	31	
90 –91	90	144–145	144	55	85	55	85	Hardware interrupt vector IRQ
	91		145	E4	228	E4	228	
92 –93	92	146–147	146	78	120	78	120	BRK interrupt vector
	93		147	D4	212	D4	212	
94 –95	94	148–149	148	FF	255	FF	255	NMI interrupt vector
	95		149	B3	179	B3	179	

Location				Contents				Description
Hex		Dec		4000 Hex	4000 Dec	8000 Hex	8000 Dec	
96	96	150	150	00	0	00	0	Status word ST
97	97	151	151	FF	255	FF	25	Which key down: 255 = no key
98	98	152	152	00	0	00	0	Shift key: 1 if depressed
99 -9A	99	153-154	153	19	25	D5	213	Correction clock
	9A		154	02	2	00	0	
9B	9B	155	155	FF	255	FF	255	Keyswitch PIA: STOP and RVS flags
9C	9C	156	156	00	0	00	0	Timing constant for tape
9D	9D	157	157	00	0	00	0	Load = 0, Verify = 1
9E	9E	158	158	00	0	00	0	# of chars in keyboard buffer
9F	9F	159	159	00	0	00	0	Screen reverse flag
A0	A0	160	160	FF	255	FF	255	IEEE output: 255 = character pending
A1	A1	161	161	1E	30	20	32	End-of-line-for-input pointer
A2	A2	162	162	00	0	00	0	Not used
A3 -A4	A3	163-164	163	0A	10	0A	10	Cursor log (row, column)
	A4		164	1E	30	20	32	
A5	A5	165	165	1E	30	20	32	IEEE output buffer
A6	A6	166	166	FF	255	FF	255	Key image
A7	A7	167	167	01	1	01	1	0 = flash cursor
A8	A8	168	168	02	2	02	2	Cursor timing countdown
A9	A9	169	169	20	32	20	32	Character under cursor
AA	AA	170	170	00	0	00	0	Cursor in blink phase
AB	AB	171	171	00	0	00	0	EOT received from tape
AC	AC	172	172	00	0	00	0	Input from screen/from keyboard
AD	AD	173	173	00	0	00	0	X save
AE	AE	174	174	00	0	00	0	How many open files
AF	AF	175	175	00	0	00	0	Input device, normally 0
B0	B0	176	176	03	3	03	3	Output CMD device, normally 3
B1	B1	177	177	00	0	00	0	Tape character parity
B2	B2	178	178	00	0	00	0	Byte received flag
B3	B3	179	179	00	0	00	0	Logical address temporary save
B4	B4	180	180	07	7	07	7	Tape buffer character; MLM command
B5	B5	181	181	00	0	00	0	File name pointer; MLM flag, counter
B6	B6	182	182	00	0	00	0	Function not known
B7	B7	183	183	00	0	00	0	Serial bit count
B8	B8	184	184	00	0	00	0	Unused
B9	B9	185	185	00	0	00	0	Cycle counter
BA	BA	186	186	00	0	00	0	Tape writer countdown
BB -BC	BB	187-188	187	00	0	00	0	Tape buffer pointers, #1 and #2
	BC		188	00	0	00	0	
BD	BD	189	189	00	0	00	0	Write leader count; read pass1/2
BE	BE	190	190	00	0	00	0	Write new byte; read error flag
BF	BF	191	191	00	0	00	0	Write start bit; read bit seq error
C0 -C1	C0	192-193	192	00	0	00	0	Error log pointers, pass1/2
	C1		193	00	0	00	0	
C2	C2	194	194	00	0	00	0	0 = scan/1-15 = count/$40 = load $80 = end
C3	C3	195	195	00	0	00	0	Write leader length; read checksum
C4 -C5	C4	196-197	196	90	144	20	32	Pointer to screen line
	C5		197	81	129	83	131	
C6	C6	198	198	1E	31	21	33	Position of cursor on above line
C7 -C8	C7	199-200	199	C7	199	C7	199	Utility pointer: tape, scroll
	C8		200	00	0	00	0	
C9 -CA	C9	201-202	201	00	0	24	36	Tape end addrs/end of current prog
	CA		202	01	1	10	16	
CB -CC	CB	203-204	203	00	0	00	0	Tape timing constants
	CC		204	00	0	00	0	
CD	CD	205	205	00	0	00	0	0 = direct cursor, else programmed
CE	CE	206	206	00	0	00	0	Tape read timer 1 = enabled
CF	CF	207	207	00	0	00	0	EOT received from tape
D0	D0	208	208	00	0	00	0	Read character error
D1	D1	209	209	0D	13	0F	15	# characters in file name
D2	D2	210	210	00	0	00	0	Current file logical address
D3	D3	211	211	61	97	61	97	Current file secondary address
D4	D4	212	212	08	8	08	8	Current file device number
D5	D5	213	213	27	39	4F	79	Right-hand window or line margin
D6 -D7	D6	214-215	214	00	0	00	0	**Pointer:** Start of Tape Buffer
	D7		215	00	0	00	0	
D8	D8	216	216	0A	10	0A	10	Line where cursor lives
D9	D9	217	217	0D	13	0D	13	Last key/checksum/misc.
DA-DB	DA	218-219	218	09	9	09	9	File name pointers
	DB		219	02	2	02	2	
DC	DC	220	220	00	0	00	0	Number of INSERTs outstanding
DD	DD	221	221	00	0	00	0	Write shift word/read character in
DE	DE	222	222	00	0	00	0	Tape blocks remaining to write/read
DF	DF	223	223	00	0	00	0	Serial word buffer
E0 -F8	E0	224-248	224	80	128			(40-column) screen wrap table
	E1		225	80	128			
	E2		226	80	128			
	E3		227	80	128			
	E4		228	80	128			
	E5		229	80	128			
	E6		230	80	128			
	E7		231	81	129			
	E8		232	81	129			
	E9		233	81	129			
	EA		234	81	129			
	EB		235	81	129			
	EC		236	81	129			
	ED		237	82	130			
	EE		238	82	130			
	EF		239	82	130			
	F0		240	82	130			
	F1		241	82	130			
	F2		242	82	130			
	F3		243	82	130			
	F4		244	83	131			
	F5		245	83	131			
	F6		246	83	131			
	F7		247	83	131			
	F8		248	83	131			
E0	E0	224	224			00	0	(80 column) Screen top window
E1	E1	225	225			18	24	(80 column) Screen bottom window
E2	E2	226	226			00	0	(80 column) Left window margin
E3	E3	227	227			09	9	(80 column) Limit of keyboard buffer
E4	E4	228	228			00	0	(80 column) Key repeat flag
E5	E5	229	229			0E	14	(80 column) Repeat countdown
E6	E6	230	230			10	16	(80 column) New key marker
E7	E7	231	231			10	16	(80 column) Chime time
E8	E8	232	232			00	0	(80 column) HOME count
E9 -EA	E9	233-234	233			1D	29	(80 column) Input vector
	EA		234			E1	225	
EB -EC	EB	235-236	235			0C	12	(80 column) Output vector
	EC		236			E2	226	
ED-F7	ED	237-247	237			00	0	(80 column) Not used
	EE		238			00	0	
	EF		239			00	0	
	F0		240			00	0	
	F1		241			00	0	
	F2		242			00	0	
	F3		243			00	0	
	F4		244			00	0	
	F5		245			00	0	
	F6		246			00	0	
	F7		247			00	0	
F8	F8	248	248			00	0	(80 column) Counter to speed TI by 6/5
F9 -FA	F9	249-250	249	00	0	00	0	Cassette status, #1 and #2
	FA		250	00	0	00	0	
FB -FC	FB	251-252	251	00	0	00	0	MLM pointer/tape start address
	FC		252	00	0	00	0	
FD-FE	FD	253-254	253	00	0	24	36	MLM, DOS pointer, misc.
	FE		254	01	1	10	16	
FF	FF	255	255	00	0	00	0	Unused

0100-010A	256-266	STR$ work area/MLM work
0100-013E	256-318	Tape read error log
0100-01FF	256-511	Processor stack
0200-0250	512-592	MLM work area; Input buffer
0251-025A	593-602	File logical address table
025B-0264	603-612	File device number table
0265-026E	613-622	File secondary addr table
026F-0278	623-632	Keyboard input buffer
027A-0339	634-825	Tape#1 input buffer
033A-03F9	826-1017	Tape#2 input buffer
033A	826	DOS character pointer
033B	827	DOS drive 1 flag
033C	828	DOS drive 2 flag
033D	829	DOS length/write flag
033E	830	DOS syntax flags
033F-0340	831-832	DOS disk ID
0341	833	DOS command string count
0342-0352	834-850	DOS file name buffer

0353-0380	851-896	DOS command string buffer
03EE-03F7	1006-1015	(80-column) Tab stop table
03FA-03FB	1018-1019	Monitor extension vector
03FC	1020	IEEE timeout defeat* $FF - disable
0400-7FFF	1024-32767	Available RAM including expansion
8000-83FF	32768-33791	(40-column) Video RAM
8000-87FF	32768-34815	(80-column) Video RAM
9000-AFFF	36864-45055	Available ROM expansion area* (2.0: -BFFF, -49151)
B000-DFFF	45056-57343	Basic, DOS, Machine Lang Monitor (2.0: Basic, C000-E0F8, 49152-57592)
E000-E7FF	57344-59391	Screen, Keyboard, Interrupt programs (2.0: E0F9-)
E810-E813	59408-59411	PIA 1 - Keyboard I/O
E820-E823	59424-59427	PIA 2 - IEEE-488 I/O
E840-E84F	59456-59471	VIA - I/O and timers
E880-E881	59520-59521	(80-column) CRT Controller
F000-FFFF	61440-65535	Reset, I/O handlers, Tape routines

VIC 20 / Commodore 64 Memory Map

Jim Butterfield, Toronto Ont.

There are some differences between the 20 and 64 as indicated. Zero Page contents at power-up by Richard Evers.

Location				Contents				Description
Hex		Dec		VIC Hex	VIC Dec	C64 Hex	C64 Dec	
00 -02	00	0-2	0	4C	76	2F	47	USR Jump. 64: Chip directional reg.
	01		1	48	72	37	55	64: Chip I/O; memory & tape control
	02		2	D2	210	33	51	20: JMP $D248. 64: Unused
03 -04	03	3-4	3	AA	170	AA	170	Float-Fixed vector
	04		4	D1	209	B1	177	
05 -06	05	5-6	5	91	145	91	145	Fixed-Float vector
	06		6	D3	211	B3	179	
07	07	7	7	22	34	22	34	Search character
08	08	8	8	22	34	22	34	Scan-quotes flag
09	09	9	9	00	0	00	0	TAB column save
0A	0A	10	10	00	0	00	0	0 = LOAD, 1 = VERIFY
0B	0B	11	11	4C	76	4C	76	Input buffer pointer/# subscripts
0C	0C	12	12	00	0	00	0	Default DIM flag
0D	0D	13	13	00	0	00	0	Type: FF = string, 00 = numeric
0E	0E	14	14	00	0	00	0	Type: 80 = integer, 00 = floating pt
0F	0F	15	15	00	0	00	0	DATA scan/LIST quote/memory flag
10	10	16	16	00	0	00	0	Subscript/FNx flag
11	11	17	17	00	0	00	0	0 = INPUT;$40 = GET;$98 = READ
12	12	18	18	00	0	00	0	ATN sign/Comparison eval. flag
13	13	19	19	05	5	05	5	Current I/O prompt flag
14 -15	14	20-21	20	14	20	14	20	Integer value
	15		21	00	0	00	0	
16	16	22	22	19	25	19	25	**Pointer:** Temporary string stack
17 -18	17	23-24	23	16	22	16	22	Last temp string vector
	18		24	00	0	00	0	
19 -21	19	25-33	25	02	25	02	2	Stack for temporary strings
	1A		26	FE	254	FE	254	
	1B		27	1D	29	9F	159	
	1C		28	0	0	00	0	
	1D		29	00	0	00	0	
	1E		30	00	0	00	0	
	1F		31	00	0	1E	30	
	20		32	00	0	00	0	
	21		33	00	0	00	0	
22 -25	22	34-37	34	05	5	05	5	Utility pointer area
	23		35	10	16	08	8	
	24		36	F3	243	F3	243	
	25		37	01	1	01	1	
26 -2A	26	38-42	38	00	0	00	0	Product area for multiplication
	27		39	00	0	00	0	
	28		40	00	0	00	0	
	29		41	00	0	00	0	
	2A		42	00	0	00	0	
2B -2C	2B	43-44	43	01	1	01	1	**Pointer:** Start of BASIC
	2C		44	10	16	08	8	
2D -2E	2D	45-46	45	03	3	03	3	**Pointer:** Start of Variables
	2E		46	10	16	08	8	
2F -30	2F	47-48	47	0A	10	0A	10	**Pointer:** Start of Arrays
	30		48	10	16	08	8	
31 -32	31	49-50	49	0A	10	0A	10	**Pointer:** End of Arrays
	32		50	10	16	08	8	
33 -34	33	51-52	51	00	0	00	0	**Pointer:** String Storage (moving down)
	34		52	1E	30	A0	160	
35 -36	35	53-54	53	00	0	00	0	**Pointer:** String Utility
	36		54	1E	30	A0	160	
37 -38	37	55-56	55	00	0	00	0	**Pointer:** Limit of Memory
	38		56	1E	30	A0	160	
39 -3A	39	57-58	57	00	0	00	0	Current BASIC line number
	3A		58	FF	255	FF	255	
3B -3C	3B	59-60	59	00	0	00	0	Previous BASIC line number
	3C		60	00	0	00	0	
3D -3E	3D	61-62	61	3D	61	00	0	**Pointer:** BASIC statement for CONT
	3E		62	00	0	00	0	
3F -40	3F	63-64	63	00	0	00	0	Current DATA line number
	40		64	00	0	00	0	
41 -42	41	65-66	65	00	0	00	0	Current DATA address
	42		66	10	16	08	8	
43 -44	43	67-68	67	00	0	00	0	Input vector
	44		68	00	0	00	0	
45 -46	45	69-70	69	41	65	41	65	Current variable name
	46		70	00	0	00	0	
47 -48	47	71-72	71	05	5	05	5	Current variable address
	48		72	10	16	08	8	
49 -4A	49	73-74	73	05	5	05	5	Variable pointer for FOR/NEXT
	4A		74	10	16	08	8	
4B -4C	4B	75-76	75	00	0	00	0	Y-save; op-save; BASIC pointer save
	4C		76	00	0	00	0	
4D	4D	77	77	00	0	00	0	Comparison symbol accumulator
4E -53	4E	78-83	78	00	0	00	0	Misc. work area, pointers, etc.
	4F		79	00	0	00	0	
	50		80	00	0	00	0	
	51		81	00	0	00	0	

Location				Contents				Description
Hex		Dec		VIC Hex	VIC Dec	C64 Hex	C64 Dec	
	52		82	00	0	00	0	
	53		83	03	3	03	3	
54 -56	54	84-86	84	4C	76	4C	76	Jump vector for functions
	55		85	0D	13	0D	13	
	56		86	D8	216	B8	184	
57 -60	57	87-96	87	00	0	00	0	Misc. numeric work area
	58		88	0A	10	0A	10	
	59		89	1F	15	07	7	
	5A		90	03	3	03	3	
	5B		91	1F	15	07	7	
	5C		92	00	0	00	0	
	5D		93	00	0	00	0	
	5E		94	00	0	00	0	
	5F		95	03	3	03	3	
	60		96	10	16	08	8	
61	61	97	97	87	135	87	135	Accum#1: Exponent
62 -65	62	98-101	98	00	0	00	0	Accum#1: Mantissa
	63		99	00	0	00	0	
	64		100	00	0	00	0	
	65		101	65	101	65	101	
66	66	102	102	4C	76	4C	76	Accum#1: Sign
67	67	103	103	00	0	00	0	Series evaluation constant pointer
68	68	104	104	00	0	00	0	Accum#1 hi-order (overflow)
69 -6E	69	105-110	105	00	0	00	0	Accum#2: Exponent
	6A		106	00	0	00	0	Accum#2: Mantissa
	6B		107	00	0	00	0	
	6C		108	00	0	00	0	
	6D		109	00	0	00	0	
	6E		110	00	0	00	0	Accum#2: Sign
6F	6F	111	111	00	0	00	0	Sign comparison, Acc#1 vs #2
70	70	112	112	00	0	00	0	Accum#1 lo-order (rounding)
71 -72	71	113-114	113	01	1	01	1	Cassette buff len/Series pointer
	72		114	01	1	01	1	
73 -8A	73	115-138	115	E6	230	E6	230	CHRGET subroutine; get BASIC char
	74		116	7A	122	7A	122	;INC $7A
	75		117	D0	208	D0	208	;BNE $0079
	76		118	02	2	02	2	
	77		119	E6	230	E6	230	;INC $7B
	78		120	7B	123	7B	123	
	79		121	AD	173	AD	173	;LDA $022D 64: LDA $022C
	7A		122	2D	45	2C	44	
	7B		123	02	2	02	2	
	7C		124	C9	201	C9	201	;CMP #$3A
	7D		125	3A	58	3A	58	
	7E		126	B0	176	B0	176	;BCS $008A
	7F		127	0A	10	0A	10	
	80		128	C9	201	C9	201	;CMP #$20
	81		129	20	32	20	32	
	82		130	F0	240	F0	240	;BEQ $0073
	83		131	EF	239	EF	239	
	84		132	38	56	38	56	;SEC
	85		133	E9	233	E9	233	;SBC #$30
	86		134	30	48	30	48	
	87		135	38	56	38	56	;SEC
	88		136	E9	233	E9	233	;SBC #$D0
	89		137	D0	208	D0	208	
	8A		138	60	96	60	96	;RTS
7A -7B	7A	122-123	122	2D	45	2C	44	BASIC pointer (within subrtn)
	7B		123	02	2	02	2	
8B -8F	8B	139-143	139	80	128	80	128	RND seed value
	8C		140	4F	79	4F	79	
	8D		141	C7	199	C7	199	
	8E		142	52	82	52	82	
	8F		143	58	88	58	88	
90	90	144	144	00	0	00	0	Status word ST
91	91	145	145	FF	255	FF	255	Keyswitch PIA: STOP and RVS flags
92	92	146	146	00	0	00	0	Timing constant for tape
93	93	147	147	00	0	00	0	LOAD = 0, VERIFY = 1
94	94	148	148	55	85	55	85	Serial output: deferred char flag
95	95	149	149	FF	255	FF	255	Serial deferred character
96	96	150	150	00	0	00	0	Tape EOT received
97	97	151	151	10	16	00	0	Register save
98	98	152	152	01	1	01	1	How many open files
99	99	153	153	00	0	00	0	Input device, normally 0
9A	9A	154	154	08	8	08	8	Output CMD device, normally 3
9B	9B	155	155	00	0	00	0	Tape character parity
9C	9C	156	156	00	0	00	0	Byte-received flag
9D	9D	157	157	80	128	80	128	Direct = $80/RUN = 0 output control
9E	9E	158	158	00	0	00	0	Tp Pass 1 error log/char buffer
9F	9F	159	159	00	0	00	0	Tp Pass 2 err log corrected
A0 -A2	A0	160-162	160	00	0	00	0	Jiffy Clock HML
	A1		161	25	37	3B	59	

Location				Contents				Description
Hex		Dec		VIC Hex	VIC Dec	C64 Hex	C64 Dec	
	A2		162	74	116	38	56	
A3	A3	163	163	55	85	55	85	Serial bit count/EOI flag
A4	A4	164	164	00	0	00	0	Cycle count
A5	A5	165	165	00	0	00	0	Countdown, tape write/bit count
A6	A6	166	166	00	0	00	0	Tape buffer pointers
A7	A7	167	167	00	0	00	0	Tp Wrt ldr count/Rd pass/inbit
A8	A8	168	168	00	0	00	0	Tp Wrt new byte/Rd error/inbit cnt
A9	A9	169	169	00	0	00	0	Wrt start bit/Rd bit err/stbit
AA	AA	170	170	00	0	00	0	Tp Scan;Cnt;Ld;End/byte assy
AB	AB	171	171	00	0	00	0	Wr lead length/Rd checksum/parity
AC–AD	AC	172–173	172	00	0	00	0	**Pointer:** tape bufr, scrolling
	AD		173	00	0	00	0	
AE–AF	AE	174–175	174	00	0	00	0	Tape end adds/End of program
	AF		175	00	0	00	0	
B0–B1	B0	176–177	176	00	0	00	0	Tape timing constants
	B1		177	00	0	00	0	
B2–B3	B2	178–179	178	3C	60	3C	60	**Pointer:** Start of Tape Buffer
	B3		179	03	3	03	3	
B4	B4	180	180	00	0	00	0	1 = Tp timer enabled; bit count
B5	B5	181	181	00	0	00	0	Tp EOT/RS232 next bit to send
B6	B6	182	182	00	0	00	0	Read character error/outbyte buf
B7	B7	183	183	11	17	10	16	# characters in file name
B8	B8	184	184	05	5	05	5	Current logical file
B9	B9	185	185	65	101	65	101	Current secndy address
BA	BA	186	186	08	8	08	8	Current device
BB–BC	BB	187–188	187	EF	239	F0	240	Pointer to file name
	BC		188	1D	29	9F	159	
BD	BD	189	189	00	0	00	0	Wr shift word/Rd input char
BE	BE	190	190	00	0	00	0	# blocks remaining to Wr/Rd
BF	BF	191	191	00	0	00	0	Serial word buffer
C0	C0	192	192	00	0	00	0	Tape motor interlock
C1–C2	C1	193–194	193	00	0	00	0	I/O start address
	C2		194	20	32	A0	160	
C3–C4	C3	195–196	195	6D	109	30	48	Kernal setup pointer
	C4		196	FD	253	FD	253	
C5	C5	197	197	40	64	40	64	Last key pressed
C6	C6	198	198	00	0	00	0	# chars in keybd buffer
C7	C7	199	199	00	0	00	0	Screen reverse flag
C8	C8	200	200	4A	74	49	73	End-of-line for input pointer
C9–CA	C9	201–202	201	04	4	03	3	Input cursor log (row, column)
	CA		202	4A	74	49	73	
CB	CB	203	203	40	64	40	64	Which key: 64 if no key
CC	CC	204	204	01	1	01	1	0 = flash cursor
CD	CD	205	205	0D	13	11	17	Cursor timing countdown
CE	CE	206	206	20	32	20	32	Character under cursor
CF	CF	207	207	00	0	00	0	Cursor in blink phase
D0	D0	208	208	00	0	00	0	Input from screen/from keyboard

Location				Contents				Description
Hex		Dec		VIC Hex	VIC Dec	C64 Hex	C64 Dec	
D1–D2	D1	209–210	209	C6	198	40	64	Pointer to screen line
	D2		210	1E	30	05	5	
D3	D3	211	211	00	0	00	0	Position of cursor on above line
D4	D4	212	212	00	0	00	0	0 = direct cursor, else programmed
D5	D5	213	213	15	21	27	39	Current screen line length
D6	D6	214	214	09	9	08	8	Row where cursor lives
D7	D7	215	215	0D	13	0D	13	Last inkey/checksum/buffer
D8	D8	216	216	00	0	00	0	# of INSERTS outstanding
D9–F0	D9	217–240	217	9E	158	84	132	Screen line link table
	DA		218	9E	158	84	132	
	DB		219	9E	158	84	132	
	DC		220	9E	158	84	132	
	DD		221	9E	158	84	132	
	DE		222	9E	158	84	132	
	DF		223	1E	30	84	132	
	E0		224	1E	30	05	5	
	E1		225	1E	30	85	133	
	E2		226	9E	158	85	133	
	E3		227	9E	158	85	133	
	E4		228	9E	158	85	133	
	E5		229	9F	159	85	133	
	E6		230	9F	159	86	134	
	E7		231	9F	159	86	134	
	E8		232	9F	159	86	134	
	E9		233	9F	159	86	134	
	EA		234	9F	159	86	134	
	EB		235	9F	159	86	134	
	EC		236	9F	159	86	134	
	ED		237	9F	159	87	135	
	EE		238	9F	159	87	135	
	EF		239	9F	159	87	135	
	F0		240	9F	159	87	135	
F1	F1	241	241	FF	255	87	135	Dummy screen link
F2	F2	242	242	08	8	87	135	Screen row marker
F3–F4	F3	243–244	243	6E	110	F0	240	Screen colour pointer
	F4		244	96	150	D8	216	
F5–F6	F5	245–246	245	5E	94	81	129	Keyboard pointer
	F6		246	EC	236	EB	235	
F7–F8	F7	247–248	247	00	0	00	0	RS-232 Rcv pntr
	F8		248	00	0	00	0	
F9–FA	F9	249–250	249	00	0	00	0	RS-232 Tx pntr
	FA		250	00	0	00	0	
FB	FB	251	251	00	0	00	0	Not Known
FC	FC	252	252	00	0	00	0	Not Known
FD	FD	253	253	00	0	00	0	Not Known
FE	FE	254	254	00	0	00	0	Not Known
FF	FF	255	255	00	0	20	32	Start of Floating to ASCII Work Area

00FF–010A	256–266	Floating to ASCII work area
0100–013E	256–318	Tape error log
0100–01FF	256–511	Processor stack area
0200–0258	512–600	BASIC input buffer
0259–0262	601–610	Logical file table
0263–026C	611–620	Device number table
026D–0276	621–630	Sec address table
0277–0280	631–640	Keybd buffer
0281–0282	641–642	Start of BASIC Memory
0283–0284	643–644	Top of BASIC Memory
0285	645	Serial bus timeout flag
0286	646	Current colour code
0287	647	Colour under cursor
0288	648	Screen memory page
0289	649	Max size of keybd buffer
028A	650	Repeat all keys
028B	651	Repeat speed counter
028C	652	Repeat delay counter
028D	653	Keyboard Shift/Control flag
028E	654	Last shift pattern
028F–0290	655–656	Keyboard table setup pntr
0291	657	Keyboard shift mode
0292	658	0 = scroll enable
0293	659	RS-232 control reg
0294	660	RS-232 command reg

		* Commodore 64 only
0295–0296	661–662	Bit timing
0297	663	RS-232 status
0298	664	# bits to send
0299–029A	665–666	RS-232 speed/code
029B	667	RS232 receive pointer
029C	668	RS232 input pointer
029D	669	RS232 transmit pointer
029E	670	RS232 output pointer
029F–02A0	671–672	IRQ save during tape I/O
02A1	673	CIA 2 (NMI) Interrupt control *
02A2	674	CIA 1 Timer A control log *
02A3	675	CIA 1 Interrupt log *
02A4	676	CIA 1 Timer A enabled flag *
02A5	677	Screen row marker *
02C0–02FE	704–766	(Sprite 11) *
0300–0301	768–769	Error message link
0302–0303	770–771	BASIC warm start link
0304–0305	772–773	Crunch BASIC tokens link
0306–0307	774–775	Print tokens link
0308–0309	776–777	Start new BASIC code link
030A–030B	778–779	Get arithmetic element link
030C	780	SYS A-reg save *
030D	781	SYS X-reg save *
030E	782	SYS Y-reg save *

030F	783	SYS status reg save	*
0310–0312	784–785	USR function jump	64: (B248) *
0314–0315	788–789	Hardware interrupt vector	20: (EABF) 64: (EA31)
0316–0317	790–791	Break interrupt vector	20: (FED2) 64: (FE66)
0318–0319	792–793	NMI interrupt vector	20: (FEAD) 64: (FE47)
031A–031B	794–795	OPEN vector	20: (F40A) 64: (F34A)
031C–031D	796–797	CLOSE vector	20: (F34A) 64: (F291)
031E–031F	798–799	Set-input vector	20: (F2C7) 64: (F20E)
0320–0321	800–801	Set-output vector	20: (F309) 64: (F250)
0322–0323	802–803	Restore I/O vector	20: (F3F3) 64: (F333)
0324–0325	804–805	INPUT vector	20: (F20E) 64: (F157)
0326–0327	806–807	Output vector	20: (F27A) 64: (F1CA)
0328–0329	808–809	Test-STOP vector	20: (F770) 64: (F6ED)
032A–032B	810–811	GET vector	20: (F1F5) 64: (F13E)
032C–032D	812–813	Abort I/O vector	20: (F3EF) 64: (F32F)
032E–032F	814–815	Warm start vector	64: (FE66) *
032E–032F	814–815	USR vector	20: (FED2)
0330–0331	816–817	LOAD link	20: (F549) 64: (F4A5)
0332–0333	818–819	SAVE link	20: (F685) 64: (F5ED)
033C–03FB	828–1019	Cassette buffer	
0340–037E	832–894	(Sprite 13)	*
0380–03BE	896–958	(Sprite 14)	*
03C0–03FE	960–1022	(Sprite 15)	*

		VIC 20
0400–0FFF	1024–4095	3K RAM expansion area
1000–1FFF	4096–8191	Normal BASIC memory
1E00–1FF9	7680–8185	Normal Screen memory
1000–11F9	4096–4601	Screen memory w/expansion
1200–	4608–	BASIC memory w/expansion
2000–7FFF	8192–32767	Memory expansion area
8000–8FFF	32768–36863	Character bit maps
9000–900F	36864–36879	Video Interface Chip
9110–912F	37136–37151	VIA Interface - NMI
9120–912F	37152–37167	VIA Interface - IRQ
9400–95FF	37888–38399	Alternate Colour Nybble area
9600–97FF	38400–38911	Main Colour Nybble area
A000–BFFF	40960–49151	Plug-in ROM area
C000–FFFF	49152–65535	ROM: BASIC and Operating System
FF8A–FFF5	65418–65525	Jump Table (Kernal)

		Commodore 64
0400–07F7	1024–2039	Screen memory (default)
07F8–07FF	2040–2047	Sprite Pointers (default)
0800–9FFF	2048–40959	BASIC RAM memory
8000–9FFF	32768–40959	Alternate: ROM plug-in area
A000–BFFF	40960–49151	ROM: BASIC
A000–BFFF	49060–49151	Alternate: RAM
C000–CFFF	49152–53247	RAM memory, including alternate
D000–D02E	53248–53294	Video Chip (6566)
D400–D41C	54272–54300	Sound Chip (6581 SID)
D800–DBFF	55296–56319	Color nybble memory
DC00–DC0F	56320–56335	Interface chip 1, IRQ (6526 CIA)
DD00–DD0F	56576–56591	Interface chip 2, NMI (6526 CIA)
D000–DFFF	53248–53294	Alternate: Character set
E000–FFFF	57344–65535	ROM: Operating System
E000–FFFF	57344–65535	Alternate: RAM
FF81–FFF5	65409–65525	Jump Table (Kernal)

BASIC 2.0 / BASIC 4.0 Memory Map

Jim Butterfield, Toronto, Ont.

Reference to DOS, MLM, 80–Column, or those marked with an * are for BASIC 4.0 only. There are some differences between usage for the 40– and 80–column machines.

Hex	Dec	Description
0000 – 0002	0–2	USR jump
0003	3	Search character
0004	4	Scan–between–quotes flag
0005	5	Input buffer pointer; # of subscripts
0006	6	Default DIM flag
0007	7	Type: FF = string, 00 = numeric
0008	8	Type: 80 = integer, 00 = floating point
0009	9	Flag: DATA scan; LIST quote; memory
000A	10	Subscript flag; FNX flag
000B	11	0 = INPUT; $40 = GET; $98 = READ
000C	12	ATN sign/Comparison Evaluation flag
000D – 000F	13–15	Disk status DS$ descriptor
0010	16	Current I/O device for prompt–suppress
0011 – 0012	17–18	Integer value (for SYS, GOTO etc)
0013 – 0015	19–21	Pointers for descriptor stack
0016 – 001E	22–30	Descriptor stack(temp strings)
001F – 0022	31–34	Utility pointer area
0023 – 0027	35–39	Product area for multiplication
0028 – 0029	40–41	**Pointer:** Start of BASIC
002A – 002B	42–43	**Pointer:** Start of Variables
002C – 002D	44–45	**Pointer:** Start of Arrays
002E – 002F	46–47	**Pointer:** End of Arrays
0030 – 0031	48–49	**Pointer:** String Storage (moving down)
0032 – 0033	50–51	**Pointer:** Utility String
0034 – 0035	52–53	**Pointer:** Limit of Memory
0036 – 0037	54–55	Current BASIC line number
0038 – 0039	56–57	Previous BASIC line number
003A – 003B	58–59	**Pointer:** BASIC statement for CONT
003C – 003D	60–61	Current DATA line number
003E – 003F	62–63	Current DATA address
0040 – 0041	64–65	Input vector
0042 – 0043	66–67	Current variable name
0044 – 0045	68–69	Current variable address
0046 – 0047	70–71	Variable pointer for FOR/NEXT
0048 – 0049	72–73	Y–save; op–save; BASIC pointer save
004A	74	Comparison symbol accumulator
004B – 0050	75–80	Misc work area, pointers, etc
0051 – 0053	81–83	Jump vector for functions
0054 – 005D	84–93	Misc numeric work area
005E	94	Accum#1: Exponent
005F – 0062	95–98	Accum#1: Mantissa
0063	99	Accum#1: Sign
0064	100	Series evaluation constant pointer
0065	101	Accum#1 hi–order (overflow)
0066 – 006B	102–107	Accum#2: Exponent, etc.
006C	108	Sign comparison, Acc#1 vs #2
006D	106	Accum#1 lo–order (rounding)
006E – 006F	110–111	Cassette buff len/Series pointer
0070 – 0087	112–135	CHRGET subroutine: get BASIC char
0077 – 0078	119–120	BASIC pointer (within subrtn)
0088 – 008C	136–140	Random number seed.
008D – 008F	141–143	Jiffy clock for TI and TI$
0090 – 0091	144–145	Hardware interrupt vector
0092 – 0093	146–147	BRK interrupt vector
0094 – 0095	148–149	NMI interrupt vector
0096	150	Status word ST
0097	151	Which key down; 255 = no key
0098	152	Shift key: 1 if depressed
0099 – 009A	153–154	Correction clock
009B	155	Keyswitch PIA: STOP and RVS flags
009C	156	Timing constant for tape
009D	157	Load = 0, Verify = 1
009E	158	Number of characters in keybd buffer
009F	159	Screen reverse flag
00A0	160	IEEE output; 255 = character pending
00A1	161	End–of–line–for–input pointer
00A3 – 00A4	163–164	Cursor log (row, column)
00A5	165	IEEE output buffer
00A6	166	Key image
00A7	167	0 = flash cursor
00A8	168	Cursor timing countdown
00A9	169	Character under cursor
00AA	170	Cursor in blink phase
00AB	171	EOT received from tape
00AC	172	Input from screen/from keyboard
00AD	173	X save
00AE	174	How many open files
00AF	175	Input device, normally 0
00B0	176	Output CMD device, normally 3
00B1	177	Tape character parity
00B2	178	Byte received flag
00B3	179	Logical Address temporary save
00B4	180	Tape buffer character/MLM command
00B5	181	File name pointer/MLM flag, counter
00B7	183	Serial bit count
00B9	185	Cycle counter
00BA	186	Tape writer countdown
00BB – 00BC	187–188	Tape buffer pointers, #1 and #2
00BD	189	Write leader count; read pass1/2
00BE	190	Write new byte; read error flag
00BF	191	Write start bit; read bit seq error
00C0 – 00C1	192–193	Error log pointers, pass1/2
00C2	194	0 = Scan/1–15 = Count/$40 = Load/$80 = End
00C3	195	Write leader length; read checksum
00C4 – 00C5	196–197	Pointer to screen line
00C6	198	Position of cursor on above line
00C7 – 00C8	199–200	Utility pointer: tape, scroll
00C9 – 00CA	201–202	Tape end addrs/End of current program
00CB – 00CC	203–204	Tape timing constants
00CD	205	0 = direct cursor, else programmed
00CE	206	Tape read timer 1 enabled
00CF	207	EOT received from tape
00D0	208	Read character error
00D1	209	# characters in file name
00D2	210	Current file logical address
00D3	211	Current file secondary addrs
00D4	212	Current file device number
00D5	213	Right–hand window or line margin
00D6 – 00D7	214–215	**Pointer:** Start of Tape Buffer
00D8	215	Line where cursor lives
00D9	217	Last key/checksum/misc.
00DA – 00DB	218–219	File name pointer
00DC	220	Number of INSERTs outstanding
00DD	221	Write shift word/read character in
00DE	222	Tape blocks remaining to write/read
00DF	223	Serial word buffer
00E0 – 00F8	224–248	(40–column) Screen line wrap table
00E0 – 00E1	224–225	(80–column) Top, bottom of window
00E2	226	(80–column) Left window margin
00E3	227	(80–column) Limit of keybd buffer
00E4	228	(80–column) Key repeat flag
00E5	229	(80–column) Repeat countdown
00E6	230	(80–column) New key marker
00E7	231	(80–column) Chime time
00E8	232	(80–column) HOME count
00E9 – 00EA	233–234	(80–column) Input vector
00EB – 00EC	235–236	(80–column) Output vector
00F9 – 00FA	249–250	Cassette status, #1 and #2
00FB – 00FC	251–252	Tape start address/MLM Pointer
00FD – 00FE	253–254	MLM/DOS pointer/misc.
0100 – 010A	256–266	STR$ work area/MLM work
0100 – 013E	256–318	Tape read error log
0100 – 01FF	256–511	Processor stack
0200 – 0250	512–592	MLM work area; Input buffer
0251 – 025A	593–602	File logical address table
025B – 0264	603–612	File device number table
0265 – 026E	613–622	File secondary addr table
026F – 0278	623–632	Keyboard input buffer
027A – 0339	634–825	Tape#1 input buffer
033A – 03F9	826–1017	Tape#2 input buffer
033A	826	DOS character pointer
033B	827	DOS drive 1 flag
033C	828	DOS drive 2 flag
033D	829	DOS length/write flag
033E	830	DOS syntax flags
033F – 0340	831–832	DOS disk ID
0341	833	DOS command string count
0342 – 0352	834–850	DOS file name buffer
0353 – 0380	851–896	DOS command string buffer
03EE – 03F7	1006–1015	(80–column) Tab stop table
03FA – 03FB	1018–1019	Monitor extension vector
03FC	1020	IEEE timeout defeat* $FF–disable
0400 – 7FFF	1024–32767	Available RAM including expansion
8000 – 83FF	32768–33791	(40–column) Video RAM
8000 – 87FF	32768–34815	(80–column) Video RAM
9000 – AFFF	36864–45055	Available ROM expansion area* (2.0: –BFFF, –49151)
B000 – DFFF	45056–57343	BASIC, DOS, Machine Lang Monitor (2.0 BASIC: C000–E0F8, 49152–57592)
E000 – E7FF	57344–59391	Screen, Keyboard, Interrupt programs (2.0: E0F9–)
E810 – E813	59408–59411	PIA 1 – Keyboard I/O
E820 – E823	59424–59427	PIA 2 – IEEE–488 I/O
E840 – E84F	59456–59471	VIA – I/O and timers
E880 – E881	59520–59521	(80–column) CRT Controller
F000 – FFFF	61440–65535	Reset, I/O handlers, Tape routines

E810	Diagnostic Sense	IEEE EOI In	Cassette Sense #2	Cassette Sense #1	Keyboard Row Select			PA	59408
E811	Tape #1 Input Flag		EOI Out	CA2	DDRA Access	Cassette #1 Read Control	CA1		59409
E812	Keyboard Row Input								59410
E813	Retrace I Flag		Cassette #1 Motor Output	CB2	DDRB Access	Retrace Interrupt Control	CB1		58411

E820	IEEE Input								59424
E821	ATN I Flag		IEEE NDAC Out	CA2	DDRA Access	IEEE ATN In Control	CA1		59425
E822	IEEE Output								59426
E823	SRQ I Flag		IEEE DAV Out	CB2	DDRB Access	IEEE SRQ In Control	CB1		59427

E840	DAV In	NRFD In	Retrace In	Cass. #2 Motor	Cassette Output	ATN Out	NRFD Out	NDAC In	PB 59456
E841	Parallel User Port (PUP) I/O with Handshake								59457
E842	Data Direction Register B (for E840)								59458
E843	Data Direction Register A (for E84F, PUP.)								59459
E844	Timer 1							L	59460
E845								H	59461
E846	Timer 1 Latch							L	59462
E847								H	59463
E848	Timer 2							L	59464
E849								H	59465
E84A	Shift Register								59466
E84B	T1 Control PB7 Out	one-shot free-run	T2 Ctrl PB6 Sense	Shift Register Control			PB, PA Latch Control		59467
E84C	CB2 (PUP, Pin M) Control In/Out			CB1 In, Cassette #2 Polarity	CA2 (Graphics, Lower Case) In/Out			CA1 In Polarity	59468
E84D	IRQ Status	T1 INT	T2 INT	CB1 Cass #2 INT	CB2 INT	SRQ INT	CA1 (PUPB) INT	CA2 INT	59469
E84E	Enable Clear/Set	T1 INT Enab	T2 INT Enab	CB1 INT Enab	CB2 INT Enab	SRQ INT Enab	CA1 INT Enab	CA2 INT Enab	59470
E84F	Parallel User Port I/O (PA)							PA	59471

BASIC 2.0 / BASIC 4.0 ROM Routines

Jim Butterfield, Toronto Ont.

The BASIC 4.0 40-character and 80-character machines are the same except for addresses $E000–$E7FF. This map shows where various routines lie. The first address is not necessarily the proper entry point for the routine. Similarly, many routines require register setup or data preparation before calling.

BASIC 2.0 ROM Routines

Address	Description
C000 – C045	Action addresses for primary keywords
C046 – C073	Action addresses for functions
C074 – C091	Hierarchy & action addrs for operators
C092 – C19Z	Table of BASIC keywords
C193 – C2A8	BASIC messages, mostly error msgs
C2AA – C2D7	Search stack FOR/GOSUB
C2D8 – C31A	Open up space in memory
C31B – C327	Test: stack too deep?
C328 – C354	Check available memory
C355	Send canned error message, then:
C389 – C3AA	Warm start (ready.)
C3AB – C441	Handle new BASIC line input
C442 – C46E	Rebuild chaining of BASIC lines
C46F – C494	Receive line from keyboard
C495 – C52B	Crunch keywords into BASIC tokens
C52C – C55A	Search BASIC for given line number
C55B	Perform [NEW], and:
C577 – C5A6	Perform [CLR]
C5A7 – C5B4	Reset BASIC execution to start
C5B5 – C657	Perform [LIST]
C658 – C6FF	Perform [FOR]
C700 – C72F	Execute BASIC statement
C730 – C73E	Perform [RESTORE]
C73F – C76A	Perform [STOP] or [END]
C76B – C784	Perform [CONT]
C785 – C78F	Perform [RUN]
C790 – C7AC	Perform [GOSUB]
C7AD – C7D9	Perform [GOTO]
C7DA	Perform [RETURN], then:
C7F3 – C80D	Perform [DATA]: skip statement
C80E	Scan for next BASIC statement
C811 – C82F	Scan for next BASIC line
C830	Perform [IF], and perhaps:
C843 – C852	Perform [REM]: skip line
C853 – C872	Perform [ON]
C873 – C8AC	Accept fixed-point number
C8AD – C8BA	Perform [LET]
C8BB – C90Z	Perform [PRINT#]
C991 – C9A4	Perform [CMD]
C9A5 – CA1B	Perform [PRINT]
CA1C – CA38	Print string from memory
CA39 – CA4E	Print single format character
CA4F – CA7C	Handle bad input data
CA7D – CAA6	Perform [GET]
CAA7 – CAC0	Perform [INPUT#]
CAC1 – CAF9	Perform [INPUT]
CAFA – CB06	Prompt and receive input
CB07 – CBFB	Perform [READ]
CBFC – CC1F	Canned Input error messages
CC20 – CC78	Perform [NEXT]
CC79 – CC9E	Check type mismatch
CC9F	Evaluate expression
CDEC	Evaluate expr. within ( )
CDF2 – CE02	Check parenthesis, comma
CE03 – CE07	Syntax error exit
CE08 – CE88	Variable name setup
CE89 – CEC7	Set up function references
CEC8 – CEF7	Perform [OR], [AND]
CEF8 – CF5F	Perform comparisons
CF60 – CF6C	Perform [DIM]
CF6D – CFF6	Search for variable
CFF7 – D077	Create new variable
D078 – D088	Setup array pointer
D089 – D08C	32768 in floating binary
D08D – D0AB	Evaluate integer expression
D0AC – D258	Find or make array
D259	Perform [FRE], and:
D26D – D279	Convert fixed-to-floating
D27A – D27F	Perform [POS]
D280 – D28C	Check not Direct
D28D – D2BA	Perform [DEF]
D2BB – D2CD	Check FNx syntax
D2CE – D33E	Evaluate FNx
D33F – D34E	Perform [STR$]
D34F – D360	Do string vector
D361 – D3CD	Scan, set up string
D3CE – D3FF	Allocate space for string
D400 – D516	Garbage collection
D517 – D553	Concatenate
D554 – D57C	Store string
D57D – D5B4	Discard unwanted string
D5B5 – D5C5	Clean descriptor stack
D5C6 – D5D9	Perform [CHR$]
D5DA – D605	Perform [LEFT$]
D606 – D610	Perform [RIGHT$]
D611 – D63A	Perform [MID$]
D63B – D655	Pull string data
D656 – D65B	Perform [LEN]
D65C – D664	Switch string to numeric
D665 – D674	Perform [ASC]
D675 – D686	Get byte parameter
D687 – D6C5	Perform [VAL]
D6C6 – D6D1	Parameters for POKE/WAIT
D6D2 – D6E7	Convert floating-to-fixed
D6E8 – D706	Perform [PEEK]
D707 – D70F	Perform [POKE]
D710 – D728	Perform [WAIT]
D72C – D732	Add 0.5
D733 – D744	Perform subtraction
D745 – D76D	Microsoft Joke (WAIT 6502)
D76E – D852	Perform addition
D853 – D889	Complement accum#1
D88A – D88E	Overflow exit
D88F – D8C7	Multiply-a-byte
D8C8 – D8E5	Constants
D8F6	Perform [LOG]
D937 – D997	Perform multiplication
D998 – D9C2	Unpack memory into accum#2
D9C3 – D9DF	Test & adjust accumulators
D9E0 – D9ED	Handle overflow and underflow
D9EE – DA04	Multiply by 10
DA05 – DA09	10 in floating binary
DA0A	Divide by 10
DA13	Perform divide-by
DA1E – DAAD	Perform divide-into
DAAE – DAD2	Unpack memory into accum#1
DAD3 – DB07	Pack accum#1 into memory
DB08 – DB17	Move accum#2 to #1
DB18 – DB26	Move accum#1 to #2
DB27 – DB36	Round accum#1
DB37 – DB44	Get accum#1 sign
DB45 – DB53	Perform [SGN]
DB54 – DB56	Perform [ABS]
DB67 – DBA6	Compare accum#1 to memory
DBA7 – DBD7	Floating-to-fixed
DBD8 – DBFE	Perform [INT]
DBFF – DC89	Convert string to floating-point
DC8A – DCBE	Get new ASCII digit
DCBF – DCCD	Constants
DCCE	Print IN, then:
DCD9 – DCE8	Print BASIC line #
DCE9 – DE1C	Convert floating-point to ASCII
DE1D – DE5D	Constants
DE5E	Perform [SQR]
DE68	Perform power function
DEA1 – DEAB	Perform negation
DEAC – DED9	Constants
DEDA – DF2C	Perform [EXP]
DF2D – DF76	Series evaluation
DF77 – DF7E	RND constants
DF7F – DFD7	Perform [RND]
DFD8	Perform [COS]
DFDF – E027	Perform [SIN]
E028 – E053	Perform [TAN]
E054 – E08B	Constants
E08C – E0BB	Perform [ATN]
E0BC – E0F8	Constants
E0F9 – E110	CHRGET sub for zero page
E111 – E115	Initial RND seed
E116 – E1B6	BASIC cold start
E1B7 – E1DD	Power up msg, "bytes free"
E1DE	Init I/O regs and:
E229	Clear screen and:
E257 – E284	Home cursor
E285 – E33E	Input from screen or keyboard
E33F – E34B	Test for quote; test quote flag
E34C – E38A	Set screen print parameters
E38B – E395	Prevent 80-char line getting longer
E396 – E3B3	Turn 40 char line into 80 char line
E3B4 – E3D7	Back into previous line
E3D8 – E518	Handle ASCII char for screen output
E519 – E53E	Go to next screen line
E53F – E5B9	Scroll screen
E5BA – E61A	Open a line on screen
E61B – E62D	Main Interrupt entry
E62E – E6E9	Interrupt: clock, cursor, keyboard
E6EA – E6F7	Output character
E5F8 – E769	Table: keyboard matrix decoder
E76A – E796	MLM sub: output hex digits
E797 – E7A5	MLM sub: swap TMP0 and TMP2
E7A7 – E7F6	MLM sub: input hex digits
E7F7 – E7FF	MLM sub: print '?'
F000 – F0B5	File messages
F0B6 – F127	Send 'Talk', 'Listen', IEEE command
F128 – F135	Send char to IEEE
F136 – F155	Write Timeout, Device Not Present
F156 – F163	Send canned I/O message
F164 – F16E	Send 'Listen', secondary address
F16F – F17E	Send normal (deferred) IEEE char
F17F – F188	Drop IEEE device
F18C – F1D0	Input byte from IEEE
F1D1 – F1E0	GET a byte
F1E1 – F231	INPUT a byte
F232 – F26D	Output a byte
F26E	Abort files
F284 – F28C	Restore default I/O devices
F28D – F2A8	Find/setup file data
F2A9 – F300	Perform [CLOSE]
F301 – F30E	Test STOP key
F30F – F314	Action STOP key
F315 – F31C	Send message if Direct mode
F31D – F321	Test if Direct mode
F322 – F3C1	Program load subroutine
F3C2 – F405	Perform [LOAD]
F40A – F43D	Print Searching, Loading, Verifying
F43E – F45F	Get Load/Save parameters
F460 – F465	Get a byte parameter
F466 – F493	Send filename to IEEE
F494 – F4B6	Find specific tape header
F4B7 – F4CD	Perform [VERIFY]
F4CE – F50D	Get Open/Close parameters
F50E – F515	Abort if end-of-line
F516 – F520	Check comma, else Syntax Error
F521 – F5A5	Perform [OPEN]
F5A6 – F5D9	Find any tape header
F5DA – F638	Write tape header
F63C – F655	Get start/end addrs from header
F656 – F66B	Set buffer address
F66C – F683	Set buffer start & end addrs
F684 – F68C	Perform [SYS]
F68D – F69D	Set tape write start & end
F69E – F728	Perform [SAVE]
F729 – F76C	Update clock
F76D – F76F	TI constant: modulus 24 hrs
F770 – F7B8	Connect input device
F7BC – F805	Connect output device
F806 – F811	Bump tape buffer pointer
F812 – F834	Wait for PLAY
F835 – F846	Test cassette switches
F847 – F854	Wait for PLAY AND RECORD
F855	Initiate tape read
F886 – F8E5	Initiate tape write
F8E6 – F8EF	Test I/O complete
F8F0 – F8FF	Test STOP key
F900 – F930	Tape bit timing adjust
F931	Read tape bits
FA57 – FB75	Store tape characters
FB76 – FB7E	Reset tape read address
FB7F – FB83	Flag error into ST
FB84 – FB92	Reset counters for new byte
FB93 – FBAE	Write a bit to tape
FBAF – FC40	Tape write
FC41 – FC7A	Write tape leader
FC7B – FC95	Terminate tape; restore interrupt
FC96 – FCA5	Set interrupt vector
FCA6 – FCB3	Turn off tape motor
FCB4 – FCC5	Checksum calculation
FCC6 – FCD0	Advance load/save pointer
FCD1 – FCFD	Power-on Reset
FCFE – FD10	Table of interrupt vectors
FD11 – FFB0	Machine Language Monitor
FFB1 – FFBF	CBM copyright statement
Jump Table:	
FFC0	OPEN
FFC3	CLOSE
FFC6	Set input device
FFC9	Set output device
FFCC	Restore default I/O devices
FFCF	INPUT a byte
FFD2	Output a byte
FFD5	LOAD
FFD8	SAVE
FFDB	VERIFY
FFDE	SYS
FFE1	Test stop key
FFE4	GET byte
FFE7	Abort all files
FFEA	Update clock
FFED – FFF9	Unused
FFFA – FFFF	Hard vectors: NMI, Reset, IRQ

BASIC 4.0 ROM Routines

Address	Description
B000 – B065	Action addresses for primary keywords
B066 – B093	Action addresses for functions
B094 – B0B1	Hierarchy & action addrs for operators
B0B2 – B20C	Table of BASIC keywords
B20D – B321	BASIC messages, mostly error msgs
B322 – B34F	Search stack for FOR/GOSUB
B350 – B392	Open up space in memory
B393 – B39F	Test: stack too deep?
B3A0 – B3CC	Check available memory
B3CD	Send canned error message, then:
B3FF – B41E	Warm start; wait for BASIC command
B41F – B4B5	Handle new BASIC line input
B4B6 – B4E1	Rebuild chaining of BASIC lines
B4E2 – B4FA	Receive line from keyboard
B4FB – B5A2	Crunch keywords into BASIC tokens
B5A3 – B5D1	Search BASIC for given line number
B5D2	Perform [NEW], and:
B5EC – B621	Perform [CLR]
B622 – B62F	Reset BASIC execution to start
B630 – B6DD	Perform [LIST]
B6DE – B784	Perform [FOR]
B785 – B7B6	Execute BASIC statement
B7B7 – B7C5	Perform [RESTORE]
B7C6 – B7ED	Perform [STOP] or [END]
B7EE – B807	Perform [CONT]
B808 – B812	Perform [RUN]
B813 – B82F	Perform [GOSUB]
B830 – B86C	Perform [GOTO]
B85D	Perform [RETURN], then:
B883 – B890	Perform [DATA]: skip statement
B891	Scan for next BASIC statement
B894 – B8B2	Scan for next BASIC line
B8B3	Perform [IF], and perhaps:
B8C6 – B8D5	Perform [REM]: skip line
B8D6 – B8F5	Perform [ON]
B8F6 – B92F	Accept fixed-point number
B930 – BA87	Perform [LET]
BA88 – BA8D	Perform [PRINT#]
BA8E – BAA1	Perform [CMD]
BAA2 – BB1C	Perform [PRINT]
BB1D – BB39	Print string from memory
BB3A – BB4B	Print single format character
BB4C – BB79	Handle bad input data
BB7A – BBA3	Perform [GET]
BBA4 – BBBD	Perform [INPUT#]
BBBE – BBF4	Perform [INPUT]
BBF5 – BC01	Prompt and receive input
BC02 – BCF6	Perform [READ]
BCF7 – BD18	Canned Input error messages
BD19 – BD71	Perform [NEXT]
BD72 – BD97	Check type mismatch
BD98	Evaluate expression
BE29	Evaluate expr. within parentheses
BE2F	Check parenthesis, comma
BF00 – BF0B	Syntax error exit
BF8C – C046	Variable name setup
C047 – C085	Set up function references
C086 – C0B5	Perform [OR], [AND]
C0B6 – C11D	Perform comparisons
C11E – C12A	Perform [DIM]
C12B – C1BF	Search for variable
C1C0 – C2C7	Create new variable
C2C8 – C2D8	Setup array pointer
C2D9 – C2DC	32768 in floating binary
C2DD – C2FB	Evaluate integer expression
C2FC – C4A7	Find or make array
C4A8	Perform [FRE], and:
C4BC – C4C8	Convert fixed-to-floating
C4C9 – C4CE	Perform [POS]
C4CF – C4DB	Check not Direct
C4DC – C509	Perform [DEF]
C50A – C51C	Check FNx syntax
C51D – C58D	Evaluate FNx
C58E – C59D	Perform [STR$]
C59E – C5AF	Do string vector
C5B0 – C61C	Scan, set up string
C51D – C669	Allocate space for string
C66A – C74E	Garbage collection
C74F – C788	Concatenate
C78C – C7B4	Store string
C7B5 – C810	Discard unwanted string
C811 – C821	Clean descriptor stack
C822 – C835	Perform [CHR$]
C836 – C861	Perform [LEFT$]
C862 – C86C	Perform [RIGHT$]
C86D – C896	Perform [MID$]
C897 – C8B1	Pull string data
C8B2 – C8B7	Perform [LEN]
C8B8 – C8C0	Switch string to numeric
C8C1 – C8D0	Perform [ASC]
C8D1 – C8E2	Get byte parameter
C8E3 – C920	Perform [VAL]
C921 – C92C	Parameters for POKE/WAIT
C92D – C942	Convert floating-to-fixed
C943 – C959	Perform [PEEK]
C95A – C962	Perform [POKE]
C963 – C97E	Perform [WAIT]
C97F – C985	Add 0.5
C986	Perform subtraction
C998 – CA7C	Perform addition
CA7D – CAB3	Complement accum#1
CAB4 – CAB8	Overflow exit
CAB9 – CAF1	Multiply-a-byte
CAF2 – CB1F	Constants
CB20	Perform [LOG]
CB5E – CBC1	Perform multiplication
CBC2 – CBEC	Unpack mem. into accum#2
CBED – CC09	Test & adjust accumulators
CC0A – CC17	Handle overflow & underflow
CC18 – CC2E	Multiply by 10
CC2F – CC33	10 in floating binary
CC34	Divide by 10
CC3D	Perform divide-by
CC45 – CCD7	Perform divide-into
CCD8 – CCFC	Unpack mem. into accum#1
CCFD – CD31	Pack accum#1 into memory
CD32 – CD41	Move accum#2 to #1
CD42 – CD50	Move accum#1 to #2
CD51 – CD60	Round accum#1
CD61 – CD6E	Get accum#1 sign
CD6F – CD8D	Perform [SGN]
CD8E – CD90	Perform [ABS]
CD91 – CDD0	Compare accum#1 to memory
CDD1 – CE01	Floating-to-fixed
CE02 – CE28	Perform [INT]
CE29 – CEB3	Convert string to floating-pt
CEB4 – CEE8	Get new ASCII digit
CEE9 – CEF8	Constants
CF78	Print IN, then:
CF7F – CF92	Print BASIC line #
CF93 – D0C6	Convert floating-pt to ASCII
D0C7 – D107	Constants
D108	Perform [SQR]
D112	Perform power function
D148 – D155	Perform negation
D156 – D183	Constants
D184 – D1D6	Perform [EXP]
D1D7 – D220	Series evaluation
D221 – D228	RND constants
D229 – D281	Perform [RND]
D282	Perform [COS]
D289 – D2D1	Perform [SIN]
D2D2 – D2FD	Perform [TAN]
D2FE – D32B	Constants
D32C – D35B	Perform [ATN]
D35C – D398	Constants
D399 – D3B5	CHRGET sub for zero page
D3B6 – D471	BASIC cold start
D472 – D716	Machine Language Monitor
D717 – D7AB	MLM subroutines
D7AC – D802	Perform [RECORD]
D803 – D837	Disk parameter checks
D838 – D872	Dummy disk control msgs
D873 – D919	[CATALOG] or [DIRECTORY]
D91A – D92E	Output
D92F – D941	Find spare secondary address
D942 – D976	Perform [DOPEN]
D977 – D990	Perform [APPEND]
D991 – D9D1	Get disk status
D9D2 – DA06	Perform [HEADER]
DA07 – DA30	Perform [DCLOSE]
DA31 – DA64	Set up disk record
DA65 – DA7D	Perform [COLLECT]
DA7E – DAA6	Perform [BACKUP]
DAA7 – DAC6	Perform [COPY]
DAC7 – DAD3	Perform [CONCAT]
DAD4 – DB0C	Insert command string values
DB0D – DB39	Perform [DSAVE]
DB3A – DB65	Perform [DLOAD]
DB66 – DB98	Perform [SCRATCH]
DB99 – DB9D	Check Direct command
DB9E – DBD6	Query ARE YOU SURE?
DBD7 – DBE0	Print BAD DISK
DBE1 – DBF9	Clear DS$ and ST
DBFA – DC67	Assemble disk command string
DC68 – DE29	Parse BASIC DOS command
DE2C – DE48	Get Device number
DE49 – DE86	Get file name
DE87 – DE9C	Get small variable parameter
**Entry points only for E000–E7FF **	
E000	Register/screen initialization
E0A7	Input from keyboard
E116	Input from screen
E202	Output character
E442	Main Interrupt entry
E455	Interrupt: clock, cursor, keyboard
E600	Exit from Interrupt
F000 – F0D1	File messages
F0D2	Send 'Talk'
F0D6	Send 'Listen'
F0D7	Send IEEE command character
F109 – F142	Send byte to IEEE
F143 – F150	Send byte and clear ATN
F151 – F16B	Option: timeout or wait
F16C – F16F	DEVICE NOT PRESENT
F170 – F184	Timeout on read, clear control lines
F185 – F192	Send canned file message
F193 – F19D	Send byte, clear control lines
F19E – F1AD	Send normal (deferred) IEEE char
F1AE – F1BF	Drop IEEE device
F1C0 – F204	Input byte from IEEE
F205 – F214	GET a byte
F215 – F265	INPUT a byte
F266 – F2A1	Output a byte
F2A2	Abort files
F2A6 – F2C0	Restore default I/O devices
F2C1 – F2DC	Find/setup file data
F2DD – F334	Perform [CLOSE]
F335 – F342	Test STOP key
F343 – F348	Action STOP key
F349 – F350	Send message if Direct mode
F351 – F355	Test if Direct mode
F356 – F400	Program load subroutine
F401 – F448	Perform [LOAD]
F449 – F46C	Print SEARCHING
F46D – F47C	Print LOADING or VERIFYING
F47D – F4A4	Get Load/Save parameters
F4A5 – F4D2	Send name to IEEE
F4D3 – F4F5	Find specific tape header
F4F6 – F50C	Perform [VERIFY]
F50D – F55F	Get Open/Close parameters
F560 – F5E4	Perform [OPEN]
F5E5 – F618	Find any tape header
F619 – F67A	Write tape header
F67B – F694	Get start/end addrs from header
F695 – F6AA	Set buffer address
F6AB – F6C2	Set buffer start & end addrs
F6C3 – F6CB	Perform [SYS]
F6CC – F6DC	Set tape write start & end
F5DD – F767	Perform [SAVE]
F768 – F7AE	Update clock
F7AF – F7FD	Connect input device
F7FE – F84A	Connect output device
F84B – F856	Bump tape buffer pointer
F857 – F879	Wait for PLAY
F87A – F888	Test cassette switch
F88C – F899	Wait for RECORD
F89A	Initiate tape read
F8CB	Initiate tape write
F8E0 – F92A	Common tape I/O
F92B – F934	Test I/0 complete
F935 – F944	Test STOP key
F945 – F975	Tape bit timing adjust
F976 – FA9B	Read tape bits
FA9C – FBBA	Read tape characters
FBBB – FBC3	Reset tape read address
FBC4 – FBC8	Flag error into ST
FBC9 – FBD7	Reset counters for new byte
FBD8 – FBF3	Write a bit to tape
FBF4 – FC85	Tape write
FC86 – FCBF	Write tape leader
FCC0 – FCDA	Terminate tape; restore interrupt
FCDB – FCEA	Set interrupt vector
FCEB – FCF8	Turn off tape motor
FCF9 – FD0A	Checksum calculation
FD0B – FD15	Advance load/save pointer
FD16 – FD4B	Power-on Reset
FD4C – FD5C	Table of interrupt vectors
Jump Table:	
FF93 – FF9E	CONCAT,DOPEN,DCLOSE,RECORD
FF9F – FFAA	HEADER,COLLECT,BACKUP,COPY
FFAB – FFB6	APPEND,DSAVE,DLOAD,CATALOG
FFB7 – FFBC	RENAME,SCRATCH
FFBD	Get disk status
FFC0	OPEN
FFC3	CLOSE
FFC6	Set input device
FFC9	Set output device
FFCC	Restore default I/O devices
FFCF	INPUT a byte
FFD2	Output a byte
FFD5	LOAD
FFD8	SAVE
FFDB	VERIFY
FFDE	SYS
FFE1	Test stop key
FFE4	GET byte
FFE7	Abort all files
FFEA	Update clock
FFED – FFF9	unused
FFFA – FFFF	Hard vectors: NMI, Reset, IRQ

VIC 20 Memory Map

Jim Butterfield, Toronto, Ont.

0000-0002	0-2	USR jump
0003-0004	3-4	Float-Fixed vector
0005-0006	5-6	Fixed-Float vector
0007	7	Search character
0008	8	Scan-quotes flag
0009	9	TAB column save
000A	10	0 = LOAD, 1 = VERIFY
000B	11	Input buffer pointer/# subscrpt
000C	12	Default DIM flag
000D	13	Type: FF = string, 00 = numeric
000E	14	Type: 80 = integer, 00 = floating point
000F	15	DATA scan/LIST quote/memry flag
0010	16	Subscript/FNx flag
0011	17	0 = INPUT;$40 = GET;$98 = READ
0012	18	ATN sign/Comparison eval flag
0013	19	Current I/O prompt flag
0014-0015	20-21	Integer value
0016	22	**Pointer:** temporary strg stack
0017-0018	23-24	Last temp string vector
0019-0021	25-33	Stack for temporary strings
0022-0025	34-37	Utility pointer area
0026-002A	38-42	Product area for multiplication
002B-002C	43-44	**Pointer:** Start of BASIC
002D-002E	45-46	**Pointer:** Start of Variables
002F-0030	47-48	**Pointer:** Start of Arrays
0031-0032	49-50	**Pointer:** End of Arrays
0033-0034	51-52	**Pointer:** String Storage (moving down)
0035-0036	53-54	**Pointer:** Utility String
0037-0038	55-56	**Pointer:** Limit of Memory
0039-003A	57-58	Current BASIC line number
003B-003C	59-60	Previous BASIC line number
003D-003E	61-62	**Pointer:** BASIC statement for CONT
003F-0040	63-64	Current DATA line number
0041-0042	65-66	Current DATA address
0043-0044	67-68	Input vector
0045-0046	69-70	Current variable name
0047-0048	71-72	Current variable address
0049-004A	73-74	Variable pointer for FOR/NEXT
004B-004C	75-76	Y-save; op-save; BASIC pointer save
004D	77	Comparison symbol accumulator
004E-0053	78-83	Misc work area, pointers, etc
0054-0056	84-86	Jump vector for functions
0057-0060	87-96	Misc numeric work area
0061	97	Accum#1: Exponent
0062-0065	98-101	Accum#1: Mantissa
0066	102	Accum#1: Sign
0067	103	Series evaluation constant pointer
0068	104	Accum#1 hi-order (overflow)
0069	105	Accum#2: Exponent
006A-006D	106-109	Accum#2: Mantissa
006E	110	Accum#2: Sign
006F	111	Sign comparison, Acc#1 vs #2
0070	112	Accum#1 lo-order (rounding)
0071-0072	113-114	Cassette buff len/Series pointer
0073-008A	115-138	CHRGET subroutine; get BASIC char
007A-007B	122-123	BASIC pointer (within subrtn)
008B-008F	139-143	RND seed value
0090	144	Status word ST
0091	145	Keyswitch PIA: STOP and RVS flags
0092	146	Timing constant for tape
0093	147	Load = 0, Verify = 1
0094	148	Serial output: deferred char flag
0095	149	Serial deferred character
0096	150	Tape EOT received
0097	151	Register save
0098	152	How many open files
0099	153	Input device, normally 0
009A	154	Output CMD device, normally 3
009B	155	Tape character parity
009C	156	Byte-received flag
009D	157	Direct = $80/RUN = 0 output control
009E	158	Tp Pass 1 error log/char buffer
009F	159	Tp Pass 2 err log corrected
00A0-00A2	160-162	Jiffy Clock HML
00A3	163	Serial bit count/EOI flag
00A4	164	Cycle count
00A5	165	Countdown,tape write/bit count
00A6	166	Tape buffer pointer
00A7	167	Tp Wrt ldr count/Rd pass/inbit
00A8	168	Tp Wrt new byte/Rd error/inbit cnt
00A9	169	Wrt start bit/Rd bit err/stbit
00AA	170	Tp Scan;Cnt;Ld;End/byte assy
00AB	171	Wr lead length/Rd checksum/parity
00AC-00AD	172-173	**Pointer:** tape bufr, scrolling
00AE-00AF	174-175	Tape end adds/End of program
00B0-00B1	176-177	Tape timing constants
00B2-00B3	178-179	**Pointer:** Start of Tape Buffer
00B4	180	1 = Tp timer enabled; bit cnt
00B5	181	Tp EOT/RS232 next bit to send
00B6	182	Read character error/outbyte buf
00B7	183	# characters in file name
00B8	184	Current logical file
00B9	185	Current secndy address
00BA	186	Current device
00BB-00BC	187-188	Pointer to file name
00BD	189	Wr shift word/Rd input char
00BE	190	# blocks remaining to Wr/Rd
00BF	191	Serial word buffer
00C0	192	Tape motor interlock
00C1-00C2	193-194	I/O start adds
00C3-00C4	195-196	Kernel setup pointer
00C5	197	Last key pressed
00C6	198	# chars in keybd buffer
00C7	199	Screen reverse flag
00C8	200	End-of-line for input pointer
00C9-00CA	201-202	Input cursor log (row, column)
00CB	203	Which key: 64 if no key
00CC	204	0 = flash cursor
00CD	205	Cursor timing countdown
00CE	206	Character under cursor
00CF	207	Cursor in blink phase
00D0	208	Input from screen/from keyboard
00D1-00D2	209-210	Pointer to screen line
00D3	211	Position of cursor on above line
00D4	212	0 = direct cursor, else programmed
00D5	213	Current screen line length
00D6	214	Row where cursor lives
00D7	215	Last inkey/checksum/buffer
00D8	216	# of INSERTs outstanding
00D9-00F0	217-240	Screen line link table
00F1	241	Dummy screen link
00F2	242	Screen row marker
00F3-00F4	243-244	Screen color pointer
00F5-00F6	245-246	Keyboard pointer
00F7-00F8	247-248	RS-232 Rcv pntr
00F9-00FA	249-250	RS-232 Tx pntr
00FF-010A	256-266	Floating to ASCII work area
0100-013E	256-318	Tape error log
0100-01FF	256-511	Processor stack area
0200-0258	512-600	BASIC input buffer
0259-0262	601-610	Logical file table
0263-026C	611-620	Device # table
026D-0276	621-630	Sec Adds table
0277-0280	631-640	Keybd buffer
0281-0282	641-642	Start of BASIC Memory
0283-0284	643-644	Top of BASIC Memory
0285	645	Serial bus timeout flag
0286	646	Current colour code
0287	647	Colour under cursor
0288	648	Screen memory page
0289	649	Max size of keybd buffer
028A	650	Repeat all keys
028B	651	Repeat speed counter
028C	652	Repeat delay counter
028D	653	Keyboard Shift/Control flag
028E	654	Last shift pattern
028F-0290	655-656	Keyboard table setup pointer
0291	657	Keymode (Kattacanna)
0292	658	0 = scroll enable
0293	659	VIC chip control
0294	660	VIC chip command
0295-0296	661-662	Bit timing
0297	663	RS-232 status
0298	664	# bits to send
0299-029A	665-666	RS-232 speed/code
029B	667	RS232 receive pointer
029C	668	RS232 input pointer
029D	669	RS232 transmit pointer
029E	670	RS232 output pointer
029F-02A0	671-672	IRQ save during tape I/O
0300-0301	768-769	Error message link
0302-0303	770-771	BASIC warm start link
0304-0305	772-773	Crunch BASIC tokens link
0306-0307	774-775	Print tokens link
0308-0309	776-777	Start new BASIC code link
030A-030B	778-779	Get arithmetic element link
030C-0313	780-787	Unused
0314-0315	788-789	Hardware interrupt vector (EABF)
0316-0317	790-791	Break interrupt vector (FED2)
0318-0319	792-793	NMI interrupt vector (FEAD)
031A-031B	794-795	OPEN vector (F40A)
031C-031D	796-797	CLOSE vector (F34A)
031E-031F	798-799	Set-input vector (F2C7)
0320-0321	800-801	Set-output vector (F309)
0322-0323	802-803	Restore I/O vector (F3F3)
0324-0325	804-805	INPUT vector (F20E)
0326-0327	806-807	Output vector (F27A)
0328-0329	808-809	Test-STOP vector (F770)
032A-032B	810-811	GET vector (F1F5)
032C-032D	812-813	Abort I/O vector (F3EF)
032E-032F	814-815	USR vector (FED2)
0330-0331	816-817	LOAD link (F549)
0332-0333	818-819	SAVE link (F685)
033C-03FB	828-1019	Cassette buffer
03FC-03FF	1020-1023	Unused
0400-0FFF	1024-4095	3K RAM expansion area
1000-1DFF	4096-7679	Normal BASIC memory
1E00-1FF9	7680-8191	Normal Screen memory
1000-11F9	4096-4601	Screen memory w/expansion
1200-	4608-	BASIC memory w/expansion
2000-7FFF	8192-32767	Memory expansion area
8000-8FFF	32768-36863	Character bit maps
9000-900F	36864-36879	Video Interface Chip
9110-912F	37136-37151	VIA interface – NMI
9120-912F	37152-37167	VIA interface – IRQ
9400-95FF	37888-38399	Alternate Colour Nybble area
9600-97FF	38400-38911	Main Colour Nybble area
A000-BFFF	40960-49151	Plug-in ROM area
C000-FFFF	49152-65535	ROM: BASIC and Operating System
FF8A-FFF5	65418-65525	Jump Table, Including
FFC6		Set Input channel
FFC9		Set Output channel
FFCC		Restore delault I/O channels
FFCF		INPUT
FFD2		PRINT
FFE1		Test Stop key
FFE4		GET

VIC 20 ROM Routines

C000	ROM control vectors
C00C	Keyword action vectors
C052	Function vectors
C080	Operator vectors
C09E	Keywords
C19E	Error messages
C328	Error message vectors
C365	Miscellaneous messages
C38A	Scan stack for FOR/GOSUB
C3B8	Move memory
C3FB	Check stack depth
C408	Check memory space
C435	'OUT OF MEMORY'
C437	Error routine
C469	Break entry
C474	'READY.'
C480	Ready for BASIC
C49C	Handle new line
C533	Re-chain lines
C560	Receive input line
C579	Crunch tokens
C613	Find BASIC line
C642	Perform [NEW]
C65E	Perform [CLR]
C68E	Back up text pointer
C69C	Perform [LIST]
C742	Perform [FOR]
C7ED	Execute statement
C81D	Perform [RESTORE]
C82C	Break
C82F	Perform [STOP]
C831	Perform [END]
C857	Perform [CONT]
C871	Perform [RUN]
C883	Perform [GOSUB]
C8A0	Perform [GOTO]
C8D2	Perform [RETURN]
C8F8	Perform [DATA]
C906	Scan for next statement
C928	Perform [IF]
C93B	Perform [REM]
C94B	Perform [ON]
C96B	Get fixed point number
C9A5	Perform [LET]
CA80	Perform [PRINT#]
CA86	Perform [CMD]
CAA0	Perform [PRINT]
CB1E	Print message from (y,a)
CB3B	Print format character
CB4D	Bad-input routines
CB7B	Perform [GET]
CBA5	Perform [INPUT#]
CBBF	Perform [INPUT]
CBF9	Prompt & input
CC06	Perform [READ]
CCFC	Input error messages
CD1E	Perform [NEXT]
CD78	Type-match check
CD9E	Evaluate expression
CEA8	Constant - PI
CEF1	Evaluate within brackets
CEF7	Check for ')'
CEFF	Check for comma
CF08	Syntax error
CF14	Check range
CF28	Search for variable
CFA7	Set up FN reference
CFE6	Perform [OR]
CFE9	Perform [AND]
D016	Compare
D081	Perform [DIM]
D08B	Locate variable
D113	Check alphabetic
D11D	Create variable
D194	Array pointer subroutine
D1A5	Value 32768
D1B2	Float-fixed conversion
D1D1	Set up array
D245	'BAD SUBSCRIPT'
D248	'ILLEGAL QUANTITY'
D34C	Compute array size
D37D	Perform [FRE]
D391	Fixed-float conversion
D39E	Perform [POS]
D3A6	Check direct
D3B3	Perform [DEF]
D3E1	Check FN syntax
D3F4	Perform [FN]
D465	Perform [STR$]
D475	Calculate string vector
D487	Set up string
D4F4	Make room for string
D526	Garbage collection
D58D	Check salvageability
D606	Collect string
D63D	Concatenate
D67A	Build string to memory
D6A3	Discard unwanted string
D6DB	Clean descriptor stack
D6EC	Perform [CHR$]
D700	Perform [LEFT$]
D72C	Perform [RIGHT$]
D737	Perform [MID$]
D761	Pull string parameters
D77C	Perform [LEN]
D782	Exit string-mode
D78B	Perform [ASC]
D79B	Input byte parameter
D7AD	Perform [VAL]
D7EB	Get params for POKE/WAIT
D7F7	Float-fixed
D80D	Perform [PEEK]
D824	Perform [POKE]
D82D	Perform [WAIT]
D849	Add 0.5
D850	Subtract-from
D853	Perform [SUBTRACT]
D86A	Perform [ADD]
D947	Complement fac#1
D97E	'OVERFLOW'
D983	Multiply by zero byte
D9EA	Perform [LOG]
DA28	Perform [MULTIPLY]
DA59	Multiply-a-bit
DA8C	Memory to FAC#2
DAB7	Adjust FAC#1/#2
DAD4	Underflow/overflow
DAE2	Multiply by 10
DAF9	+10 in floating pt
DAFE	Divide by 10
DB12	Perform [DIVIDE]
DBA2	Memory to fac#1
DBC7	FAC#1 to memory
DBFC	FAC#2 to fac#1
DC0C	FAC#1 to FAC#2
DC1B	Round FAC#1
DC2B	Get sign
DC39	Perform [SGN]
DC58	Perform [ABS]
DC5B	Compare FAC#1 to mem
DC9B	Float-fixed
DCCC	Perform [INT]
DCF3	String to fac
DD7E	Get ASCII digit
DDDD	Float to ASCII
DF16	Decimal constants
DF3A	TI constants
DF71	Perform [SQR]
DF7B	Perform [POWER]
DFB4	Perform [NEGATIVE]
DFED	Perform [EXP]
E040	Series evaluate 1
E056	Series evaluate 2
E094	Perform [RND]
E0F6	?? Breakpoints ??
E127	Perform [SYS]
E153	Perform [SAVE]
E162	Perform [VERIFY]
E165	Perform [LOAD]
E1BB	Perform [OPEN]
E1C4	Perform [CLOSE]
E1D1	Parameters for LOAD/SAVE
E203	Check default parameters
E20B	Check for comma
E216	Parameters for open/close
E261	Perform [COS]
E268	Perform [SIN]
E2b1	Perform [TAN]
E30B	Perform [ATN]
E378	Initialize
E387	CHRGET for zero page
E3A4	Initialize BASIC
E429	Power-up message
E44F	Vectors for $300
E45B	Initialize vectors
E467	Warm restart
E476	Program patch area
E4A0	Serial output '1'
E4A9	Serial output '0'
E4B2	Get serial input & clock
E4BC	Program patch area
E500	Set 6522 addrs
E505	Set screen limits
E50A	Track cursor location
E518	Initalize I/O
E54C	Normalize screen
E55F	Clear screen
E581	Home cursor
E587	Set screen pointers
E5BB	Set I/O defaults
E5C3	Set VIC chip defaults
E5CF	Input from keyboard
E64F	Input from screen
E6B8	Quote mark test
E6C5	Set up screen print
E6EA	Advance cursor
E715	Retreat cursor
E72D	Back into previous line
E742	Output to screen
E8C3	Go to next line
E8D8	Do 'RETURN'
E8E8	Check line decrement
E8FA	Check line increment
E912	Set colour code
E921	Colour code table
E929	Code conversion
E975	Scroll screen
E9EE	Open space on screen
EA56	Move screen line
EA6E	Synch colour transfer
EA7E	Set start-of-line
EA8D	Clear screen line
EAA1	Print to screen
EAAA	Store on screen
EAB2	Synch colour to char
EABF	Interrupt (IRQ)
EB1E	Check keyboard
EC00	Set text mode
EC46	Keyboard vectors
EC5E	Keyboard maps
ED21	Graphics/text control
ED30	Set graphics mode
ED5B	Wrap up screen line
ED6A	Shifted key matrix
EDA3	Control key matrix
EDE4	VIC chip defaults
EDFD	Screen line adds low
EE14	Send 'talk'
EE17	Send 'listen'
EE1C	Send control char
EE49	Send to serial bus
EEB7	Timeout on serial
EEC0	Send listen SA
EEC5	Clear ATN
EECE	Send talk SA
EEE4	Send serial deferred
EEF6	Send 'untalk'
EF04	Send 'unlisten'
EF19	Receive from serial bus
EF84	Clock line on
EF8D	Clock line off
EF96	Delay 1 ms
EFA3	RS232 send (NMI)
EFEE	New RS232 byte send
F015	Error or quit
F027	Compute bit count
F036	RS232 receive (NMI)
F05B	Setup to receive
F09D	Receive parity error
F0A2	Receive overrun error
F0A5	Receive break error
F0A8	Receive frame error
F0E9	Bad device
F0EC	File to RS232
F0ED	Send to RS232 buffer
F116	Input from RS232 buffer
F14F	Get from RS232 buffer
F160	Check serial bus idle
F174	Messages
F1E2	Print if direct
F1F5	Get..
F205	..from RS232
F20E	Input
F250	Get.. tape/serial/RS232
F27A	Output..
F290	..to tape
F2C7	Set input device
F309	Set output device
F34A	Close
F3CF	Find file
F3DF	Set file values
F3EF	Abort all files
F3F3	Restore default I/O
F40A	Do file opening
F455	Send SA
F4C7	Open RS232
F542	LOAD program
F647	'SEARCHING'
F659	Print file name
F66A	'LOADING/VERIFYING'
F675	SAVE program
F728	'SAVING'
F734	Bump clock
F760	Get time
F767	Set time
F770	Action stop key
F77E	File Error Messages
F7AF	Find any tape header
F7E7	Write tape header
F84D	Get buffer address
F854	Set buffer start, end pointers
F867	Find specific header
F88A	Bump tape pointer
F894	'PRESS PLAY ..'
F8AB	Check cassette status
F8B7	'PRESS RECORD ..'
F8C0	Initiate tape read
F8E0	Initiate tape write
F8F4	Common tape read/write
F94B	Check tape stop
F95D	Set timing
F96E	Read bits (IRQ)
FAAD	Store characters
FBD2	Reset pointer
FBD8	New tape character setup
FBEA	Toggle tape
FC06	Data write
FC0B	Tape write (IRQ)
FC95	Leader write (IRQ)
FCCF	Restore vectors
FCF6	Set vector
FD08	Kill motor
FD11	Check read/write pointer
FD1B	Bump read/write pointer
FD22	Powerup entry
FD3F	Check A-ROM
FD52	Set Kernal2
FD8D	Initialize system constants
FDF1	IRQ vectors
FDF9	Initialize I/O regs
FE49	Save data name
FE50	Save file details
FE57	Get status
FE66	Flag ST
FE6F	Set timeout
FE73	Read/set top of memory
FE82	Read/set bottom of memory
FE91	Test memory location
FEA9	NMI interrupt entry
FED2	RESET/STOP warm start
FEDE	NMI RS232 sequences
FF56	Restore & exit
FF5C	RS232 timing table
FF72	Main IRQ entry
FF8A	Jumbo jump table
FFFA	Hardware vectors

VIC 20 Standard Configuration

FFFF	8K Kernal ROM	65535
E000	8K BASIC ROM	57344
C000		49152
A000		40960
95FF		38399
9600	Colour Nybble Area	38400
9000	VIC Chip & I/O	36864
8000	Character Set	32768
2000		4096
1E00	½K Screen RAM from basic VIC 20	7680
1000	3½ K RAM for BASIC	4096
0400		1024
0000	1K RAM Work Space	0

VIC 20 Expansion RAM Memory Changes

Exp RAM at:	BASIC Text	Screen	Colour Table
none	4096 / $1000	7680 / $1E00	38400 / $9600
1024 / 4095*	1024 / $0400	7680 / $1E00	38400 / $9600
8192 and up	4608 / $1200	4096 / $1000	37888 / $9400

* VIC 1210 3K RAM Expander

VIC 20 With 40K RAM

David Berezowski

VIC 1020 Expansion Module Required with:
1 - VIC 1210 3K RAM
2 - VIC 1110 8K RAM (Switches 2,3,4 down - Switch 1 up)
3 - VIC 1110 8K RAM (Switches 1,3,4 down - Switch 2 up)
4 - VIC 1111 16K RAM

FFFF	8K Kernal ROM	65535
E000	8K BASIC ROM	57344
C000	VIC 1110 8K RAM (2) (usable only with PEEK, POKE & M/L)	49152
A000		40960
95FF		38399
9400	Colour Nybble Area	37888
9000	VIC Chip & I/O	36864
8000	Character Set	32768
	VIC 1110 8K RAM (3)	
	VIC 1111 16K RAM (4)	27½ K for BASIC
	3½ K of RAM from basic VIC 20	
1200		4608
1000	½K Screen RAM from basic VIC 20	4096
	VIC 1210 3K RAM (1) (usable only with PEEK, POKE & M/L)	
0400		1024
0000	1K RAM Work Space	0

6560 VIC Chip

9000	Interlace	Left Margin (=5)		36864
9001	Top Margin (=25)			36865
9002	Screen Ad Bit 9	Number of Columns (=22)		36866
9003	Bit 0	Number of Rows (=23)	Double Char	36867
9004	Input Raster Value: Bits 1-8			36868
9005	Screen Address Bits 13-10	Character Address Bits 13-10		36869
9006	Light Pen Input	Horizontal		36870
9007		Vertical		36871
9008	Paddle Input	X		36872
9009		Y		36873
900A	ON	Voice 1 Frequency		36874
900B	ON	Voice 2 Frequency		36875
900C	ON	Voice 3 Frequency		36876
900D	ON	Noise Frequency		36877
900E	Multi Colour Mode	Sound Amplitude		36878
900F	Background Colour	Foregnd/ Backgnd	Border Colour	36879

6522 VIA 1

9110	DSR In	CTS In		DCD* In	RI* In	DTR Out	RTS Out	Data In	37136
	RS-232 Interface or Parallel User Port								
9111	*Unused - see $911F								37137
9112	Data Direction Register B (for $9110)								37138
9113	Data Direction Register A (for $911F)								37139
9114	T1-L	RS 232 Send Speed; Tape Write Timing							37140
9115	T1-H								37141
9116	T1-Latch L								37142
9117	T1 Latch H								37143
9118	T2-L	RS 232 Input Timing							37144
9119	T2-H								37145
911A	Shift Register (* unused)								37146
911B	T1 Control		T2 Ctrl	Shift Register Control			PB LE	PA LE	37147
911C	CB2: RS 232 Send			CB1 Ctrl	CA2: Tape Motor Ctrl			CA1 Ctrl	37148
911D	NMI:	T1	T2	CB1: RS 232 In			CA1: RESTORE		37149
911E	NMI En.	T1 Enab	T2 Enab	CB1 En.			CA1 En.		37150
911F	ATN Out	Tape Sense	Joystick Fire	Left	Down	Up	Serial Data In	Serial Clock In	37151

6522 VIA 2

9120	Joystick Right				Tape Out				37152
	Keyboard Row Select								
9121	Keyboard Column Input								37153
9122	Data Direction Register B (for $9120)								37154
9123	Data Direction Register A (for $9121)								37155
9124	T1-L	Cassette Tape Read; Keyboard and Clock Interrupt Timing							37156
9125	T1-H								37157
9126	T1-Latch L								37158
9127	T1 Latch H								37159
9128	T2-L	Serial Bus Timing							37160
9129	T2-H	Tape R/W Timing							37161
912A	Shift Register (* unused)								37162
912B	T1 Control		T2 Ctrl	Shift Register Control			PB LE	PA LE	37163
912C	Serial Bus Data Out			CB1 Ctrl	Serial Clock Line Out			CA1 Ctrl	37164
912D	IRQ:	T1	T2	CB1: SRQ In			CA1: Tape In		37165
912E	IRQ En.	T1 Enab	T2 Enab	CB1 En.			CA1 En.		37166
912F	*Unused (see $9121)								37167

Commodore 64 Memory Map

Jim Butterfield, Toronto, Ont.

0000	0	Chip directional register
0001	1	Chip I/O; memory & tape control
0003 - 0004	3-4	Float-Fixed vector
0005 - 0006	5-6	Fixed-Float vector
0007	7	Search character
0008	8	Scan-quotes flag
0009	9	TAB column save
000A	10	0= LOAD, 1= VERIFY
000B	11	Input buffer pointer/# subscrpt
000C	12	Default DIM flag
000D	13	Type: FF= string, 00= numeric
000E	14	Type: 80= integer, 00= floating point
000F	15	DATA scan/LIST quote/memry flag
0010	16	Subscript/FNx flag
0011	17	0= INPUT;$40= GET;$98= READ
0012	18	ATN sign/Comparison eval flag
0013	19	Current I/O prompt flag
0014 - 0015	20-21	Integer value
0016	22	**Pointer:** temporary string stack
0017 - 0018	23-24	Last temp string vector
0019 - 0021	25-33	Stack for temporary strings
0022 - 0025	34-37	Utility pointer area
0026 - 002A	38-42	Product area for multiplication
002B - 002C	43-44	**Pointer:** Start of BASIC
002D - 002E	45-46	**Pointer:** Start of Variables
002F - 0030	47-48	**Pointer:** Start of Arrays
0031 - 0032	49-50	**Pointer:** End of Arrays
0033 - 0034	51-52	**Pointer:** String Storage (moving down)
0035 - 0036	53-54	**Pointer:** Utility String
0037 - 0038	55-56	**Pointer:** Limit of Memory
0039 - 003A	57-58	Current BASIC line number
003B - 003C	59-60	Previous BASIC line number
003D - 003E	61-62	**Pointer:** BASIC statement for CONT
003F - 0040	63-64	Current DATA line number
0041 - 0042	65-66	Current DATA address
0043 - 0044	67-68	Input vector
0045 - 0046	69-70	Current variable name
0047 - 0048	71-72	Current variable address
0049 - 004A	73-74	Variable pointer for FOR/NEXT
004B - 004C	75-76	Y-save; op-save; BASIC pointer save
004D	77	Comparison symbol accumulator
004E - 0053	78-83	Misc work area, pointers, etc
0054 - 0056	84-86	Jump vector for functions
0057 - 0060	87-96	Misc numeric work area
0061	97	Accum#1: Exponent
0062 - 0065	98-101	Accum#1: Mantissa
0066	102	Accum#1: Sign
0067	103	Series evaluation constant pointer
0068	104	Accum#1 hi-order (overflow)
0069	105	Accum#2: Exponent, etc.
006A - 006D	106-109	Accum#2: Mantissa
006E	110	Accum#2: Exponent, etc.
006F	111	Sign comparison, Acc#1 vs #2
0070	112	Accum#1 lo-order (rounding)
0071 - 0072	113-114	Cassette buff len/Series pointer
0073 - 008A	115-138	CHRGET subroutine; get BASIC char
007A - 007B	122-123	BASIC pointer (within subrtn)
008B - 008F	139-143	RND seed value
0090	144	Status word ST
0091	145	Keyswitch PIA: STOP and RVS flags
0092	146	Timing constant for tape
0093	147	Load = 0, Verify = 1
0094	148	Serial output: deferred char flag
0095	149	Serial deferred character
0096	150	Tape EOT received
0097	151	Register save
0098	152	How many open files
0099	153	Input device, normally 0
009A	154	Output CMD device, normally 3
009B	155	Tape character parity
009C	156	Byte-received flag
009D	157	Direct = $80/RUN = 0 output control
009E	158	Tp Pass 1 error log/char buffer
009F	159	Tp Pass 2 err log corrected
00A0 - 00A2	160-162	Jiffy Clock HML
00A3	163	Serial bit count/EOI flag
00A4	164	Cycle count
00A5	165	Countdown;tape write/bit count
00A6	166	Tape buffer pointer
00A7	167	Tp Wrt ldr count/Rd pass/inbit
00A8	168	Tp Wrt new byte/Rd error/inbit cnt
00A9	169	Wrt start bit/Rd bit err/stbit
00AA	170	Tp Scan;Cnt Ld;End/byte assy
00AB	171	Wr lead length/Rd checksum/parity
00AC - 00AD	172-173	**Pointer:** tape buff, scrolling
00AE - 00AF	174-175	Tape end adds/End of program
00B0 - 00B1	176-177	Tape timing constants
00B2 - 00B3	178-179	**Pointer:** Start of Tape Buffer
00B4	180	1 = Tp timer enabled; bit count
00B5	181	Tp EOT/RS232 next bit to send
00B6	182	Read character error/outbyte buf
00B7	183	# characters in file name
00B8	184	Current logical file
00B9	185	Current secndy address
00BA	186	Current device
00BB - 00BC	187-188	Pointer to file name
00BD	189	Wr shift word/Rd input char
00BE	190	# blocks remaining to Wr/Rd
00BF	191	Serial word buffer
00C0	192	Tape motor interlock
00C1 - 00C2	193-194	I/O start address
00C3 - 00C4	195-196	Kernel setup pointer
00C5	197	Last key pressed
00C6	198	# chars in keybd buffer
00C7	199	Screen reverse flag
00C8	200	End-of-line for input pointer
00C9 - 00CA	201-202	Input cursor log (row, column)
00CB	203	Which key: 64 if no key
00CC	204	0 = flash cursor
00CD	205	Cursor timing countdown
00CE	206	Character under cursor
00CF	207	Cursor in blink phase
00D0	208	Input from screen/from keyboard
00D1 - 00D2	209-210	Pointer to screen line
00D3	211	Position of cursor on above line
00D4	212	0 = direct cursor, else programmed
00D5	213	Current screen line length
00D6	214	Row where cursor lives
00D7	215	Last inkey/checksum/buffer
00D8	216	# of INSERTs outstanding
00D9 - 00F2	217-242	Screen line link table
00F3 - 00F4	243-244	Screen colour pointer
00F5 - 00F6	245-246	Keyboard pointer
00F7 - 00F8	247-248	RS-232 Rcv pntr
00F9 - 00FA	249-250	RS-232 Tx pntr
00FF - 010A	256-266	Floating to ASCII work area
0100 - 013E	256-318	Tape error log
0100 - 01FF	256-511	Processor stack area
0200 - 0258	512-600	BASIC input buffer
0259 - 0262	601-610	Logical file table
0263 - 026C	611-620	Device # table
026D - 0276	621-630	Sec Adds table
0277 - 0280	631-640	Keyboard buffer
0281 - 0282	641-642	Start of BASIC Memory
0283 - 0284	643-644	Top of BASIC Memory
0285	645	Serial bus timeout flag
0286	646	Current colour code
0287	647	Colour under cursor
0288	648	Screen memory page
0289	649	Max size of keybd buffer
028A	650	Repeat all keys
028B	651	Repeat speed counter
028C	652	Repeat delay counter
028D	653	Keyboard Shift/Control flag
028E	654	Last shift pattern
028F - 0290	655-656	Keyboard table setup pointer
0291	657	Keyboard shift mode
0292	658	0 = scroll enable
0293	659	RS-232 control reg
0294	660	RS-232 command reg
0295 - 0296	661-662	Bit timing
0297	663	RS-232 status
0298	664	# bits to send
0299 - 029A	665-666	RS-232 speed/code
029B	667	RS232 receive pointer
029C	668	RS232 input pointer
029D	669	RS232 transmit pointer
029E	670	RS232 output pointer
029F - 02A0	671-672	IRQ save during tape I/O
02A1	673	CIA 2 (NMI) Interrupt Control
02A2	674	CIA 1 Timer A control log
02A3	675	CIA 1 Interrupt Log
02A4	676	CIA 1 Timer A enabled flag
02A5	677	Screen row marker
02C0 - 02FE	704-766	(Sprite 11)
0300 - 0301	768-769	Error message link
0302 - 0303	770-771	BASIC warm start link
0304 - 0305	772-773	Crunch BASIC tokens link
0306 - 0307	774-775	Print tokens link
0308 - 0309	776-777	Start new BASIC code link
030A - 030B	778-779	Get arithmetic element link
030C	780	SYS A-reg save
030D	781	SYS X-reg save
030E	782	SYS Y-reg save
030F	783	SYS status reg save
0310 - 0312	784-785	USR function jump (B248)
0314 - 0315	788-789	Hardware interrupt vector (EA31)
0316 - 0317	790-791	Break interrupt vector (FE66)
0318 - 0319	792-793	NMI interrupt vector (FE47)
031A - 031B	794-795	OPEN vector (F34A)
031C - 031D	796-797	CLOSE vector (F291)
031E - 031F	798-799	Set-input vector (F20E)
0320 - 0321	800-801	Set-output vector (F250)
0322 - 0323	802-803	Restore I/O vector (F333)
0324 - 0325	804-805	INPUT vector (F157)
0326 - 0327	806-807	Output vector (F1CA)
0328 - 0329	808-809	Test-STOP vector (F6ED)
032A - 032B	810-811	GET vector (F13E)
032C - 032D	812-813	Abort I/O vector (F32F)
032E - 032F	814-815	Warm start vector (FE66)
0330 - 0331	816-817	LOAD link (F4A5)
0332 - 0333	818-819	SAVE link (F5ED)
033C - 03FB	828-1019	Cassette buffer
0340 - 037E	832-894	(Sprite 13)
0380 - 03BE	896-958	(Sprite 14)
03C0 - 03FE	960-1022	(Sprite 15)
0400 - 07E7	1024-2039	Screen memory (default)
07F8 - 07FF	2040-2047	Sprite Pointers (default)
0800 - 9FFF	2048-40959	BASIC ROM memory
8000 - 9FFF	32768-40959	Alternate: ROM plug-in area
A000 - BFFF	40960-49151	ROM: BASIC
A000 - BFFF	49060-59151	Alternate: RAM
C000 - CFFF	49152-53247	RAM memory, including alternate
D000 - D02E	53248-53294	Video Chip (6566)
D400 - D41C	54272-54300	Sound Chip (6581 SID)
D800 - DBFF	55296-56319	Colour nybble memory
DC00 - DC0F	56320-56335	Interface chip 1, IRQ (6526 CIA)
DD00 - DD0F	56576-56591	Interface chip 2, NMI (6526 CIA)
D000 - DFFF	53248-53294	Alternate: Character set
E000 - FFFF	57344-65535	ROM: Operating System
E000 - FFFF	57344-65535	Alternate: RAM
FF81 - FFF5	65409-65525	Jump Table, Including
FFC6		Set Input channel
FFC9		Set Output channel
FFCC		Restore default I/O channels
FFCF		INPUT
FFD2		PRINT
FFE1		Test Stop key
FFE4		GET

Commodore 64 ROM Routines

A000	ROM control vectors
A00C	Keyword action vectors
A052	Function vectors
A080	Operator vectors
A09E	Keywords
A19E	Error messages
A328	Error message vectors
A365	Misc messages
A38A	Scan stack for FOR/GOSUB
A3B8	Move memory
A3FB	Check stack depth
A408	Check memory space
A435	'out of memory'
A437	Error routine
A469	BREAK entry
A474	'ready.'
A480	Ready for BASIC
A49C	Handle new line
A533	Re-chain lines
A560	Receive input line
A579	Crunch tokens
A613	Find BASIC line
A642	Perform [NEW]
A65E	Perform [CLR]
A68E	Back up text pointer
A69C	Perform [LIST]
A742	Perform [FOR]
A7ED	Execute statement
A81D	Perform [RESTORE]
A82C	Break
A82F	Perform [STOP]
A831	Perform [END]
A857	Perform [CONT]
A871	Perform [RUN]
A883	Perform [GOSUB]
A8A0	Perform [GOTO]
A8D2	Perform [RETURN]
A8F8	Perform [DATA]
A906	Scan for next statement
A928	Perform [IF]
A93B	Perform [REM]
A94B	Perform [ON]
A96B	Get fixed point number
A9A5	Perform [LET]
AA80	Perform [PRINT#]
AA86	Perform [CMD]
AAA0	Perform [PRINT]
AB1E	Print string from (y,a)
AB3B	Print format character
AB4D	Bad input routine
AB7B	Perform [GET]
ABA5	Perform [INPUT#]
ABBF	Perform [INPUT]
ABF9	Prompt & input
AC06	Perform [READ]
ACFC	Input error messages
AD1E	Perform [NEXT]
AD78	Type match check
AD8E	Evaluate expression
AEA8	Constant - PI
AEF1	Evaluate within brackets
AEF7	')'
AEFF	comma..
AF08	Syntax error
AF14	Check range
AF28	Search for variable
AFA7	Setup FN reference
AFE6	Perform [OR]
AFE9	Perform [AND]
B016	Compare
B081	Perform [DIM]
B08B	Locate variable
B113	Check alphabetic
B11D	Create variable
B194	Array pointer subroutine
B1A5	Value 32768
B1B2	Float-fixed convension
B1D1	Set up array
B245	'BAD SUBSCRIPT'
B248	'ILLEGAL QUANTITY'
B34C	Compute array size
B37D	Perform [FRE]
B391	Fix-float conversion
B39E	Perform [POS]
B3A6	Check direct
B3B3	Perform [DEF]
B3E1	Check FN syntax
B3F4	Perform [FN]
B465	Perform [STR$]
B475	Calculate string vector
B487	Set up string
B4F4	Make room for string
B526	Garbage collection
B5BD	Check salvageability
B606	Collect string
B63D	Concatenate
B67A	Build string to memory
B6A3	Discard unwanted string
B6DB	Clean descriptor stack
B6EC	Perform [CHR$]
B700	Perform [LEFT$]
B72C	Perform [RIGHT$]
B737	Perform [MID$]
B761	Pull string parameters
B77C	Perform [LEN]
B782	Exit string-mode
B78B	Perform [ASC]
B79B	Input byte parameter
B7AD	Perform [VAL]
B7EB	Parameters for POKE/WAIT
B7F7	Float-fixed
B80D	Perform [PEEK]
B824	Perform [POKE]
B82D	Perform [WAIT]
B849	Add 0.5
B850	Subtract-from
B853	Perform [subtract]
B86A	Perform [add]
B947	Complement FAC#1
B97E	'overflow'
B983	Multiply by zero byte
B9EA	Perform [LOG]
BA2B	Perform [multiply]
BA59	Multiply-a-bit
BA8C	Memory to FAC#2
BAB7	Adjust FAC#1/#2
BAD4	Underflow/overflow
BAE2	Multiply by 10
BAF9	+10 in floating pt
BAFE	Divide by 10
BB12	Perform [divide]
BBA2	Memory to FAC#1
BBC7	FAC#1 to memory
BBFC	FAC#2 to FAC#1
BC0C	FAC#1 to FAC#2
BC1B	Round FAC#1
BC2B	Get sign
BC39	Perform [SGN]
BC58	Perform [ABS]
BC5B	Compare FAC#1 to mem
BC9B	Float-fixed
BCCC	Perform [INT]
BCF3	String to FAC
BD7E	Get ASCII digit
BDC2	Print 'IN..'
BDCD	Print line number
BDDD	Float to ASCII
BF16	Decimal constants
BF3A	TI constants
BF71	Perform [SQR]
BF7B	Perform [power]
BFB4	Perform [negative]
BFED	Perform [EXP]
E043	Series eval 1
E059	Series eval 2
E097	Perform [RND]
E09	?? breakpoints ??
E12A	Perform [SYS]
E156	Perform [SAVE]
E165	Perform [VERIFY]
E168	Perform [LOAD]
E18E	Perform [OPEN]
E1C7	Perform [CLOSE]
E1D4	Parameters for LOAD/SAVE
E206	Check default parameters
E20E	Check for comma
E219	Parameters for open/close
E264	Perform [COS]
E26B	Perform [SIN]
E2B4	Perform [TAN]
E30E	Perform [ATN]
E37B	Warm restart
E394	Initialize
E3A2	CHRGET for zero page
E3BF	Initialize BASIC
E447	Vectors for $300
E453	Initialize vectors
E45F	Power-up message
E500	Get I/O address
E505	Get screen size
E50A	Put/get row/column
E518	Initialize I/O
E544	Clear screen
E566	Home cursor
E56C	Set screen pointers
E5A0	Set I/O defaults
E5B4	Input from keyboard
E632	Input from screen
E684	Quote test
E691	Setup screen print
E6B6	Advance cursor
E6ED	Retreat cursor
E701	Back into previous line
E716	Output to screen
E87C	Go to next line
E891	Perform <return>
E8A1	Check line decrement
E8B3	Check line increment
E8CB	Set colour code
E8DA	Colour code table
E8EA	Scroll screen
E965	Open space on screen
E9C8	Move a screen line
E9ED	Synchronize colour transfer
E9F0	Set start-of-line
E9FF	Clear screen line
EA13	Print to screen
EA24	Synchronize colour pointer
EA31	Interrupt - clock etc
EA87	Read keyboard
EB79	Keyboard select vectors
EB81	Keyboard 1 - unshifted
EBC2	Keyboard 2 - shifted
EC03	Keyboard 3 - 'comm'
EC44	Graphics/text contrl
EC4F	Set graphics/text mode
EC78	Keyboard 4
ECB9	Video chip setup
ECE7	Shift/run equivalent
ECF0	Screen ln address low
ED09	Send 'talk'
ED0C	Send 'listen'
ED40	Send to serial bus
ED82	Serial timeout
ED89	Send listen SA
EDBE	Clear ATN
EDC7	Send talk SA
EDCC	Wait for clock
EDDD	Send serial deferred
EDEF	Send 'untalk'
EDFE	Send 'unlisten'
EE13	Receive from serial bus
EE85	Serial clock on
EE8E	Serial clock off
EE97	Serial output '1'
EEA0	Serial output '0'
EEA9	Get serial in & clock
EEB3	Delay 1 ms
EEB8	RS-232 send
EF06	Send new RS-232 byte
EF2E	No-DSR error
EF31	No-CTS error
EF3B	Disable timer
EF4A	Compute bit count
EF59	RS232 receive
EF7E	Setup to receive
EFC5	Receive parity error
EFCA	Receive overflow
EFCD	Receive break
EFD0	Framing error
EFE1	Submit to RS232
F00D	No-DSR error
F017	Send to RS232 buffer
F04D	Input from RS232
F086	Get from RS232
F0A4	Check serial bus idle
F0BD	Messages
F12B	Print if direct
F13E	Get..
F14E	..from RS232
F157	Input
F199	Get.. tape/serial/rs232
F1CA	Output..
F1DD	..to tape
F20E	Set input device
F250	Set output device
F291	Close file
F30F	Find file
F31F	Set file values
F32F	Abort all files
F333	Restore default I/O
F34A	Do file open
F3D5	Send SA
F409	Open RS232
F49E	LOAD program
F5AF	'searching'
F5C1	Print filename
F5D2	'loading/verifying'
F5DD	SAVE program
F68F	Print 'SAVING'
F69B	Bump clock
F6BC	Log PIA key reading
F6DD	Get time
F6E4	Set time
F6ED	Check stop key
F6FB	Output error messages
F72D	Find any tape headr
F76A	Write tape header
F7D0	Get buffer address
F7D7	Set buffer start/end pointers
F7EA	Find specific header
F80D	Bump tape pointer
F817	'press play..'
F82E	Check tape status
F838	'press record..'
F841	Initiate tape read
F864	Initiate tape write
F875	Common tape code
F8D0	Check tape stop
F8E2	Set read timing
F92C	Read tape bits
FA60	Store tape chars
FB8E	Reset pointer
FB97	New character setup
FBA6	Send transition to tape
FBC8	Write data to tape
FBCD	IRQ entry point
FC57	Write tape leader
FC93	Restore normal IRQ
FCB8	Set IRQ vector
FCCA	Kill tape motor
FCD1	Check r/w pointer
FCDB	Bump r/w pointer
FCE2	Power reset entry
FD02	Check 8-rom
FD10	8-ROM mask
FD15	Kernal reset
FD1A	Kernal move
FD30	Vectors
FD50	Initialize system constnts
FD9B	IRQ vectors
FDA3	Initialize I/O
FDDD	Enable timer
FDF9	Save filename data
FE00	Save file details
FE07	Get status
FE18	Flag status
FE1C	Set status
FE21	Set timeout
FE25	Read/set top of memory
FE27	Read top of memory
FE2D	Set top of memory
FE34	Read/set bottom of memory
FE43	NMI entry
FE66	Warm start
FEB6	Reset IRQ & exit
FEBC	Interrupt exit
FEC2	RS-232 timing table
FED6	NMI RS-232 in
FF07	NMI RS-232 out
FF43	Fake IRQ
FF48	IRQ entry
FF81	Jumbo jump table
FFFA	Hardware vectors

6566 Video Chip
C64 Control & Miscellaneous Registers

D011		Extended Clr. Mode	Display Map	Bit Enable	Row Select	Y-Scroll		53265
D012	Raster Register							53266
D013	Light Pen Input						X	53267
D014							Y	53268

D016	x	x	Reset	Multi Colour	Column Select	X-Scroll	53270

D018	Screen VM13	VM12	VM11	VM10	Character Base CB13	CB12	CB11	x	53272
D019	IRQ	**Interrupt Sense:**			Light Pen	Spr-Spr Collision	Spr-Back Collision	Raster	53273
D01A		**Interrupt Enable:**			Light Pen	Spr-Spr Collisions	Spr-Back Collisions	Raster	53274

Colour Registers

D020	X	Exterior Colour (Border)	53280
D021	X	Background Colour #0	53281
D022	X	Background Colour #1	53282
D023	X	Background Colour #2	53283
D024	X	Background Colour #3	53284
D025	X	Sprite MultiColour #0	53285
D026	X	Sprite MultiColour #1	53286

6566 Video Chip
C64 Sprite Registers

	Sprite 0	Sprite 7		Sprite 0	Sprite 7
D000	D00E	X Position		53248	53262
D001	D00F	Y Position		53249	53263

Bit For Sprite#:

	7	6	5	4	3	2	1	0	
D010	X-Position High								53264
D015	Sprite Enable Flags								53269
D017	Y-Expand								53271
D01B	Background Priority								53275
D01C	Sprite MultiColour Mode								53276
D01D	X-Expand								53277
D01E	Interrupt: Sprite Collision								53278
D01F	Interrupt: Background Collision								53279

CIA 1 (IRQ) (6526)

$DC00	Paddle Sel A B		Joystick 0 Fire	Right	Left	Down	Up		FRA	56320
	Keyboard Row Select (inverted)									
$DC01		Joystick 1 Fire	Right	Left	Down	Up			FRB	56321
	Keyboard Column Read									
$DC02	$FF – All Output								DDRA	56322
$DC03	$00 – All Input								DDRB	56323
$DC04	Timer A								TAL	56324
$DC05									TAH	56325
$DC06	Timer B								TBL	56326
$DC07									TBH	56327
$DC0D		Tape Input			Timer Interrupt B	A			ICR	56333
$DC0E			One Shot	Out Mode	Time PB6 Out	Timer A Start			CRA	56334
$DC0F			One Shot	Out Mode	Time PB7 Out	Timer B Start			CRB	56335

CIA 2 (NMI) (6526)

$DD00	Serial IN	Clock IN	Serial OUT	Clock OUT	ATN OUT	RS-232 OUT	VIC II addr 15	VIC II addr 14	PRA 56576
$DD01	DSR IN	CTS IN		DCD* IN	RI* IN	DTR OUT	RTS OUT	RS-232 IN	PRB 56577
$DD02	$3F – Serial								DDRA 56578
$DD03	$00 – P.U.P. All Input			or		$06 – RS-232			DDRB 56579
$DD04	Timer A								TAL 56580
$DD05									TAH 56581
$DD06	Timer B								TBL 56582
$DD07									TBH 56583
$DD0D				RS-232 IN			Timer Interrupt B	A	ICR 56589
$DD0E								Timer A Start	CRA 56590
$DD0F								Timer B Start	CRB 56591

* Connected but not used by O.S.

Processor I/O Port (6510)

$0000	IN	IN	OUT	IN	OUT	OUT	OUT	OUT	DDR 0
$0001			Tape Motor	Tape Sense	Tape Write	D-ROM Switch	EF RAM Switch	AB RAM Switch	PR 1

SID (6581)

Voice 1	Voice 2	Voice 3				Voice 1	Voice 2	Voice 3
$D400	$D407	$D40E	Frequency		L	54272	54279	54286
$D401	$D408	$D40F			H	54273	54280	54287
$D402	$D409	$D410	Pulse Width		L	54274	54281	54288
$D403	$D40A	$D411	0 0 0 0		H	54275	54282	54289
$D404	$D40B	$D412	NSE PUL SAW TRI	Voice Type:	Key	54276	54283	54290
$D405	$D40C	$D413	Attack Time 2ms – 8ms	Decay Time 6ms – 24 sec		54277	54284	54291
$D406	$D40D	$D414	Sustain Level	Release Time 6ms – 24 sec		54278	54285	54292

Voices (write only)

$D415	0 0 0 0 0	L	54293
$D416	Filter Frequency	H	54294
$D417	Resonance	Filter Voices Ext V3 V2 V1	54295
$D418	V3 off HI BP LO	Master Volume	54296

Filter & Volume (write only)

$D419	Paddle X (A/D #1)	54297
$D41A	Paddle Y (A/D #2)	54298
$D41B	Noise 3 (random)	54299
$D41C	Envelope 3	54300

Sense (read only)

Note: Special Voice Features
(TEST, RING MOD, SYNC)
are omitted from the above diagram.

B Series Memory Map

Jim Butterfield, Toronto, Ont.

The following information applies to B systems released after April 1973, which contain a revised Machine Language Monitor. (If SYS 6 doesn't bring in a monitor display complete with a 'period' prompt, it's the wrong version).

Notable features as compared to previous Commodore products include:

- CHRGOT is no longer in RAM. "Wedge" type coding must be inserted at links $029E and $02A0 .. which is likely to make the job easier.
- BASIC vectors have "split" - now, for example, there are discrete "Start of Variables" and "End of Variables", distinct from End of BASIC and Start of Arrays. Three-byte vectors (including bank number) are not uncommon.
- The "Jump Table" at top of memory is still accessible and reasonbly consistent with previous Commodore products.
- Simple machine language programs will fit into the spare 1k of ROM at $0400-0800 without trouble. Large programs must be implemented either by plug-in memory (RAM or ROM) in bank 15, or placed into another bank (preferably bank 3); supplementary code will be needed to make all the coding components fit.

The following map contains BASIC addresses specific to the B256/80; references to banks 0 to 4 are also specific to that machine. Most of the map is of general usage, however.

All Banks:

0000	0	6509 Execution Register
0001	1	6509 Indirection Register

Bank 0: Unused.

Bank 1:

0002 – F000	2-61439	BASIC Program (text) RAM.
FA5E – FB00	61440-64512	Input buffer area

Bank 2:

0002 – FFFF	2-65535	BASIC Arrays in RAM.

Bank 3:

0002 – 7FFF	2-32767	Unused RAM.
8000 – FFFF	32768-65535	BASIC Variables in RAM.

Bank 4:

0002 – FBFF	2-64511	BASIC Strings (top down) in RAM
FC00 – FCFF	64512-64767	Unused RAM (descriptors?)
FD00 – FFFF	64768-65535	Current KEY definitions

Banks 5 to 14: Unused.

Bank 15:

0002 – 0004	2-4	USR Jump
0005 – 0008	5-8	TI$ Output Elements: H,M,S,T
0009 – 000B	9-11	Pointer: Print Using Format
000C	12	Search Character
000D	13	Scan-between-Quotes Flag
000E	14	Input point; * subscripts
000F	15	Catalog line counter
0010	16	Default DIM flag
0011	17	Type: 255 = string, 0 = integer
0012	18	Type: 128 = integer, 0 = fl point
0013	19	Crunch flag
0014	20	Subscript index
0015	21	Input = 0; Get = 64; Read = 152
0016 – 0019	22-25	Disk status work values
001A	26	Current IO device fr prompt suppress
001B – 001C	27-28	Integer value
001D – 001F	29-31	Descriptor stack pointers
0020 – 002B	32-43	Misc work pointer
002D – 002E	45-46	**Pointer:** Start of BASIC
002F – 0030	47-48	**Pointer:** End of BASIC
0031 – 0032	49-50	**Pointer:** Start of Variables
0033 – 0034	51-52	**Pointer:** End of Variables
0035 – 0036	53-54	**Pointer:** Start of Arrays
0037 – 0038	55-56	**Pointer:** End of Arrays
0039 – 003A	57-58	**Pointer:** Variable work
003B – 003C	59-60	**Pointer:** Bottom of Strings
003D – 003E	61-62	**Pointer:** Utility String
003F – 0041	63-65	**Pointer:** Top of String Memory
0042 – 0043	66-67	Current BASIC line number
0044 – 0045	68-69	Old BASIC line number
0046 – 0047	70-71	Old BASIC text pointer
0049 – 004A	73-74	Data line number
004B – 004C	75-76	Data text pointer
004D – 004E	77-78	Input pointer
004F – 0050	79-80	Variable name
0051 – 0053	81-83	Variable address
0054 – 0056	84-86	For-loop pointer
0057 – 0058	87-88	Text pointer save
005A	90	Comparison symbol accumulator
005B – 005D	91-93	Function location
005E – 0060	94-96	Working string vector
0061 – 0063	97-99	Function jump code
0064 – 006E	100-110	Work pointers, values
006F	111	Exponent sign
0070	112	Accum string prefix
0071	113	Accum*1: Exponent
0072 – 0075	114-117	Accum*1: Mantissa
0076	118	Accum*1: Sign
0077	119	Series Evaluation: Const pointer
0078	120	Accum*1 Hi order (overflow)
0079 – 007E	121-126	Accum*2: Ex, Man, Sign
007F	127	Sign comparison, Acc*1 vs *2
0080	128	Acc*1 Lo order (rounding)
0081 – 0084	129-132	Series, Work pointers
0085 – 0087	133-135	**Pointer:** BASIC text
0088 – 0089	136-137	Input pointer
008B – 008E	139-142	DOS parser work values
008F	143	Error type number
0090 – 0092	144-146	Pointer to file name
0093 – 0095	147-149	**Pointer:** Tape Buffer, Scrolling
0096 – 0098	150-152	Load end address/End of program
0099 – 009B	153-155	I/O start address
009C	156	Status word ST
009D	157	File name length
009E	158	Current logical file
009F	159	Current device
00A0	160	Current secondary address
00A1	161	Input device, normally 0
00A2	162	Output CMD device, normally 3
00A6 – 00A8	166-168	INBUF
00A9	169	Keyswitch PIA .. stop key,etc.
00AA	170	IEEE deferred flag
00AB	171	IEEE deferred character
00AC – 00AD	172-173	Segment transfer rts vector
00AE – 00B3	174-179	Monitor register save
00B4	180	Monitor stack pointer save
00B5	181	Monitor bank number save
00B7 – 00B8	183-184	Monitor IRQ save/pointer
00B9 – 00BA	185-186	Monitor memory pointer
00BB – 00BC	187-188	Monitor secondary pointer
00BD	189	Monitor counter
00BE	190	Monitor misc byte
00BF	191	Monitor device number
00C0 – 00C1	192-193	Prog Key Table address
00C2 – 00C3	194-195	Programmable key address
00C4 – 00C7	196-199	Pointers to change Prog Key Table
00C8 – 00C9	200-201	Pointer to screen line
00CA	202	Screen line number
00CB	203	Position of cursor on line
00CC	204	0 = text mode, else graphics mod
00CD	205	Key pressed: 255 if no key
00CE	206	Old cursor column
00CF	207	Old cursor row
00D0	208	New character flag
00D1	209	* keys in Keyboard buffer
00D2	210	Quotes Flag
00D3	211	Insert key counter
00D4	212	Cursor type flag
00D5	213	Screen line length
00D6	214	* keys in 'key' buffer
00D7	215	Key repeat delay
00D8	216	Key repeat speed
00D9 – 00DA	217-218	Temporary Variables
00DB	219	Current output character
00DC	220	Top line of current screen
00DD	221	Bottom line of screen
00DE	222	Left edge of current screen
00DF	223	Right edge of screen
00E0	224	Keys: 255 = none; 127 = key; 111 = shift
00E1	225	Key: 255 = none (no shift)
00E2 – 00E5	226-229	Line Wrap Bits
0100	256	Hex to binary staging area
0100 – 010A	256-266	Numeric to ASCII work area
0100 – 01FE	256-510	Stack area
01FF	511	Stack pointer save location
0200 – 020F	512-527	File name area
0210 – 0226	528-550	Disk command work area
0255 – 0256	597-598	Misc work values for WAIT, etc
0257	599	'Bank' value
0258	600	Output logical file (CMD)
0259	601	Sign of TAN
025A – 025D	602-605	Fickup subtn; misc work values
025E – 0276	606-630	PRINT USING working variables
0280 – 0281	640-641	Error routine link (854D)
0282 – 0283	642-643	Warm start link (85C5)
0284 – 0285	644-645	Crunch token link (88A9)
0286 – 0287	645-647	List link (89DB)
0288 – 0289	648-649	Command dispatch link (874C)
028A – 028B	650-651	Token evaluate link (960C)
028C – 028D	652-653	Expression eval link (95AF)
028E – 028F	654-655	CHRGOT link (B98E)
0290 – 0291	656-657	CHRGET vector (B994)
0292 – 0293	658-659	Float-fixed vector (B980)
0294 – 0295	660-661	Fixed-Float vector (9CA5)
0296 – 0297	662-663	Error trap vector
0298 – 0299	664-665	Error line number
029A – 029B	666-667	Error exit pointer
029C	668	Stack pointer save
029D – 029F	669-671	Temporary TRAP, DISPOSE bytes
02A0 – 02A5	672-677	Temporary INSTR$ bytes
02A6 – 02A7	678-679	Bank offset
0300 – 0301	768-769	IRQ vector (FBE9)
0302 – 0303	770-771	BRK vector (EE21)
0304 – 0305	772-773	NMI vector (FCAA)
0306 – 0307	774-775	OPEN vector (F6BF)
0308 – 0309	776-777	CLOSE vector (F5ED)
030A – 030B	778-779	Connect-input vector (F549)
030C – 030D	780-781	Connect-output vector (F5A3)
030E – 030F	782-783	Restore deflt I/O vector (F6A6)
0310 – 0311	784-785	Input vector (F49C)
0312 – 0313	786-787	Output vector (F4EE)
0314 – 0315	788-789	Stop key test vector (F96B)
0316 – 0317	790-791	GET vector (F43D)
0318 – 0319	792-793	Abort all files vector (F67F)
031A – 031B	794-795	Load vector (F746)
031C – 031D	796-797	Save vector (F84C)
031E – 031F	798-799	Monitor command vector (EE77)
0320 – 0321	800-801	Keyboard control vector (E01F)
0322 – 0323	802-803	Print control vector (E01F)
0324 – 0325	804-805	IEEE send LSA vector (F274)
0326 – 0327	806-807	IEEE send TSA vector (F280)
0328 – 0329	808-809	IEEE receive byte vector (F30A)
032A – 032B	810-811	IEEE send char vector (F297)
032C – 032D	812-813	IEEE send untalk vector (F2AB)
032E – 032F	814-815	IEEE send unlisten vector (F2AF)
0330 – 0331	816-817	IEEE send listen vector (F234)
0332 – 0333	818-819	IEEE send talk vector (F230)
0334 – 033D	820-829	File logical addresses table
033E – 0347	830-839	File device table
0348 – 0351	840-849	File secondary adds table
0352 – 0354	850-852	Bottom of system memory
0355 – 0357	853-855	Top of system memory
0358 – 035A	856-858	Bottom of user memory
035B – 035D	859-861	Top of user memory
035E	862	IEEE timeout: 0 = enabled
035F	863	0 = Load; 128 = Verify
0360	864	Number of open files
0361	865	Message mode byte
0363 – 0366	867-870	Misc register save bytes
0369	873	Timer toggle
036A – 0368	874-875	Cassette vector (dead end)
036F – 0371	879-881	Relocation start address
0375	885	Cassette motor flag (unused)
0376 – 0377	886-887	RS-232 Control, Command
037A	890	RS-232 Status
037B	891	RS-232 Handshake input
037C	892	RS-232 Input pointer
037D	893	RS-232 Arrival pointer
0380 – 0381	896-897	**Pointer:** Top of Memory
0382	898	Bank byte
0383	899	RVS flag
0384	900	Current line length
0385	901	Temp output char save
0386	902	0 = normal, 255 = auto insert
0387	903	0 = scrolling, 255 = no scroll
0388	904	Misc work byte for screen
0389	905	Index to prog key
038A	906	Scroll mode flag
038B	907	Bell mode flag
038C	908	Indirect bank save
038D – 03A0	909-928	Lengths of 'key' words
03A1 – 03AA	929-938	Bit mapped Tab stops
03AB – 03B4	939-948	Keyboard input buffer
03B5 – 03B6	949-950	'Key' word link (E91B)
03F8 – 03F9	1016-1017	Restart vector
03FA – 03FB	1018-1019	Restart test mask
0400 – 07FF	1024-2047	Free RAM (reserved for DOS)
0800 – 0FFF	2048-4095	Reserved for plug in RAM
1000 – 1FFF	4096-8191	Reserved for plug in DOS ROM
2000 – 7FFF	8192-23767	Reserved for cartridges
8000 – BFFF	32768-49151	BASIC ROM
C000 – CFFF	49152-53247	Unused
D000 – D7CF	53248-55247	Screen RAM
D800 – D801	55296-55297	Video controller 6545
DA00 – DA1C	55808-55836	Sound Interface Device 6581
DB00 – DB0F	56064-56079	Complex Interface Adaptor 6526
DC00 – DC0F	56320-56335	Complex Interface Adaptor 6526
DD00 – DD03	56576-56579	Asynchronous Comms IA 6551
DE00 – DE07	56832-56839	Tri Port Inerface Adaptor 6525
DF00 – DF07	57088-57095	Tri Port Inerface Adaptor 6525
E000 – FFFF	57344-65535	Kernal ROM

6525 Tri Port

DE00	NRFD	NDAC	EOI	DAV	ATN	RFN			56832
DE01	Sense	Cassette Motor	Out	ARB	Network Rx	Tx	SRQ	IFC	56833
DE02									56834
DE03	Data Direction Register For DE00								56835
DE04	Data Direction Register For DE01								56836
DE05	IRQ:			ACIA	IP	ALM	IEEE	PWR	56837
DE06	CB		CA: Graphics				IRQ Stack On		56838
DE07	Active Interrupt Register								56839

6525 Tri Port 2

DF00	Keyboard		57088
DF01	Select		57089
DF02	CRT Mode	Keyboard Read	57090
DF03	Data Direction Register for DF00 (out)		57091
DF04	Data Direction Register for DF01 (out)		57092
DF05	Data Direction Register for DF02 (in)		57093
DF06	Unused		57094

Commodore B128 ROM Routines

The following is a map of routines and data within the current (September 1983) version of the Commodore B128 computer. Caution: The same routines exists in the B256 but the addresses are not exactly the same.

8000	Jumps: Warm start, Cold start	8E24	Perform [DISPOSE]	9BA4	'bad subscript'	BA1E	Float-fixed conversion
8006	Mask: 'CBM8'	8E7A	Perform [PRINT*]	9BA7	'illegal quantity'	BA26	CHRGET - Get new BASIC character
800B	Reference Vectors (unused)	8E80	Perform [CMD]	9CF5	Evaluate [FRE]	BA29	CHRGOT - Get previous character
8027	Action vectors	8E9D	Perform [PRINT]	9D33	Evaluate [POS]	BA50	Numeric check
803B	Action (run etc) vectors	8F15	Perform [GET]	9D39	Fixed-float	BA5A	Set text bank
80A3	Function vectors	8F4B	Perform [INPUT*]	9D4A	Confirm not direct	BA69	Set bank from FAC
80D1	Operation vectors	8F66	Perform [INPUT]	9D57	Check direct mode	BA6E	Set bank from $60
80EF	Keywords	8FA8	Prompt & input	9E07	Evaluate [PEEK]	BA73	Set bank from $24
828F	Message vectors	8FEA	Perform [READ]	9E30	Evaluate [subtract]	BA78	Set bank 15 (system)
82E7	Messages	90E7	Perform [SYS]	9E4D	Evaluate [add]	BA7D	Set bank 4
8550	Print 'Out of memory'	910C	Perform [DIM]	9F5E	Overflow error	BA82	Set bank 2
8552	Error routine	9116	Perform [DEF]	9FCA	Evaluate [LOG]	BA87	Set bank 3
85AE	Print line number	9146	Perform [POKE]	A00B	Evaluate [multiply]	BA8C	Set bank 1 (text)
85C0	Warm start	9152	Perform [WAIT]	A0D0	+10 floating	BB82	Startup message
85F3	Handle new line	917F	Perform [KEY]	A0E9	Evaluate [divide]	BBA6	Link vectors ($0280)
86A4	Rechain lines	91BC	Perform [VERIFY]	A148	Error: 'division by zero'	BBE1	BASIC I/O with error traps:
86A3	Receive input line	91C8	Perform [LOAD]	A210	Evaluate [SGN]	BBE2	Perform BASIC Open
871F	Find BASIC line	921B	Perform [SAVE]	A22F	Evaluate [ABS]	BBE8	Perform BASIC Get
8751	Command dispatcher	9243	Perform [OPEN]	A2B1	Evaluate [INT]	BBEE	Perform BASIC Input
87DB	Peek stack for FOR/GOSUB	9257	Perform [CLOSE]	A3B4	Print numeric	BBF4	Perform BASIC output
8815	Open text space	92A1	Perform [CATALOG]	A3C3	Print canned message	BBFA	Perform BASIC connect-input
8866	Stack too deep?	936D	Perform [DOPEN]	A50D	+32768	BC00	Perform BASIC connect-output
8880	Check string space	937E	Perform [APPEND]	A537	Evaluate [SQR]	BC06	Perform BASIC Load
8890	Check BASIC space	93A9	Perform [DCLOSE]	A541	Evaluate [power]	BC0C	Perform BASIC Save
889F	Check array space	93C3	Perform [DSAVE]	A57A	Evaluate [negate]	BC12	Error on above BASIC I/O
88AB	'out of array space'	93CE	Perform [DLOAD]	A5B3	Evaluate [EXP]	BC1A	Output error message
88BF	Crunch tokens	93DE	Perform [BANK]	A659	Evaluate [RND]	E000	**Kernal:**
88D	Perform [LIST]	93EC	Perform [BSAVE]	A6A6	Evaluate [COS]	E24D	Set graphics mode
8A29	Perform [NEW]	940E	Perform [BLOAD]	A6AD	Evaluate [SIN]	E251	Set text mode
8A45	Perform [CLR]	9427	Perform [HEADER]	A6F8	Evaluate [TAN]	E260	Set up CRT controlr
8A90	'USING' characters	9464	Perform [SCRATCH]	A791	Evaluate [ATN]	E299	Output to screen
8A94	Perform [FOR]	949E	Perform [RECORD]	A7C0	Perform [PUDEF]	E306	Control key fanout
8B06	Perform [NEXT]	950A	Perform [DCLEAR]	A7DB	Evaluate [STR$]	E311	Escape key vector
8B79	Perform [RESTORE]	9513	Perform [COLLECT]	A805	Set up string descriptors	E314	Cursor up/down
8BA8	Perform [STOP]	952A	Perform [COPY]	A81F	Scan and set up string	E331	Cursor left/right
8BAA	Perform [END]	9546	Perform [CONCAT]	A8AB	Build string into memory	E344	Rvs/rvs off
8BE9	Perform [CONT]	9552	Perform [RENAME]	A8E6	Discard unwanted string	E34A	Home/clear
8C07	Perform [RUN]	9560	Perform [BACKUP]	A955	..clean descripter stack	E35A	Tab & tab set/clear
8C25	Perform [GOSUB]	9586	Patch area	AAD1	Evaluate [CHR$]	E394	Carriage return
8C42	Perform [IF]	95C1	Evaluate expression	AAEB	Evaluate [LEFT$]	E3B4	Move screen line
8C77	Perform [REM/ELSE]	95CF	Recursive entry	AB22	Evaluate [RIGHT$]	E48D	Ring bell
8C7C	Perform [GO]	96CB	Value of PI in binary	AB42	Evaluate [MID$]	E4BA	Delete numeric
8C84	Perform [GOTO]	96F8	Evaluate [NOT]	AB8E	Evaluate [LEN]	E4F5	Check start of line
8CB8	Perform [RETURN]	9724	Eval within parens	AB9D	Evaluate [ASC]	E51E	Locate line wrap
8CDF	Perform [DATA]	975A	Go for disk status	ABAE	Evaluate [VAL]	E532	Goto start of line
8CED	Next statement	986B	Evaluate [OR]	AD53	Allocate dynamic string space	E544	Goto end of line
8CF0	Next line	986E	Evaluate [AND]	ADB5	Garbage collection	E5AE	Delete/insert
8D16	Perform [TRAP]	98A8	Evaluate [COMPARE]	AF9D	Perform [DELETE]	E61C	Initiate load/run
8D2B	Perform [ON]	992C	Get var name/loc	AFF4	Get line range	E655	Escape key link
8D4E	Get fixed point number	99EF	Check alphabetic	B026	Perform <PRINT USING>	E658	Insert a line
8D8A	Perform [LET]	9AF5	Array pntr subrtn	B4B8	Reset text pointer	E66D	Delete a line
8DC4	Perform [RESUME]	9B66	Float-fixed	B4E5	Evaluate integer	E694	Erase right
				B501	Evaluate numeric	E6A9	Erase left
				B504	Check numeric mode	E6BD	Scroll up
				B506	Check string mode	E6BF	Scroll down
				B52E	Print format character	E6E3	Enable scrolling
				B53A	Print character	E6E5	Disable scrolling
				B7CB	Disk command formats	E7BE	Create new prog key

E949	Get prog key addrs			F1C3	Error messages		
E970	Escape sequence			F221	Print error message		
E979	Cancel escape seq			F230	Send 'talk'		
E985	Escape key vectors			F234	Send 'listen'		
E9B6	Set top/left			F236	Send IEEE command		
E9B8	Set bottom/right			F274	Send Listen SA		
E9BC	Set window			F277	Release ATN		
E9C7	Set full screen			F280	Send Talk SA		
E9D6	Enable bell			F283	Prepare IEEE in		
E9D8	Disable bell			F297	Send IEEE deferred		
E9DC	Set underline mode			F2AB	Send 'untalk'		
E9E6	Set flashing cursor			F2AF	Send 'unlisten'		
E9EC	Set solid cursor			F2B9	Send IEEE byte		
E9EF	Set non-flashing cursor			F30A	Receive IEEE byte		
E9F6	Reverse screen			F381	Open RS-232		
E9F9	(alternate characters)			F3C7	Convert to true ASCII		
EA05	Un-reverse screen			F3DC	Convert to PETSCII		
EA08	(normal characters)			F400	Allocate buffer		
EA20	Cancel auto insert			F4EE	Output		
EA23	Set auto insert			F549	Connect input		
EBA9	Load/run keys			F5A3	Connect output		
EBB3	Screen line addrs low			F5ED	Close file		
EBCB	Screen line addrs high			F63E	Find file LA		
EBE4	Control key vectors			F650	Set file details		
EC24	Default 'key' word lengths			F660	Find matching SA		
EC2E	Default 'key' words			F678	Search for file		
EC67	Bit masks			F67F	Abort all files		
EC6F	CRT controller setup			F6A6	Restore default I/O		
EE00	Monitor trap			F6BF	Open file		
EE09	Monitor call (60937)			F707	Open IEEE		
EE21	BRK entry			F746	Load		
EE55	Monitor reentry			F84C	Save		
EED5	Monitor vectors			F8E6	Read time of day		
EEF9	Perform [.X]	exit to BASIC		F90E	Set TOD/alarm		
EEFF	Set PC address			F939	File error entry points		
EF08	Set register address			F997	Power up reset		
EF17	Print prompt group			FAFD	Vectors		
EF1F	Print space			FB31	NMI entry		
EF22	Print question mark			FB34	Set function addrs		
EF27	Monitor prompt			FB43	Set file parameters		
EF31	Register heading			FB4A	Read status byte		
EF4C	Perform [.R]	register display		FB5A	Set message mode		
EF8F	Perform [.M]	memory display		FB5F	Log into status byte		
EFCB	Perform [.:]	register change		FB74	Set timeout		
EFE1	Perform [.V]	bank switch		FB78	Set/read top of memory		
EFEB	Perform [.U]			FB8D	Set/read bottom of memory		
EFF5	Perform [.;]	memory change		FBA2	Set page 3 vectors		
F010	Perform [.G]	go		FBD6	IRQ interrupt		
F04A	Perform [.L/.S]	load/save		FBE9	Interrupt routines		
F0F6	Print 2 hex bytes			FC9F	Wind up interrupt		
F0F8	Print hex byte			FE9D	Exsub - Bank Transfer Sequences		
F107	Print hex digit			FF84	..excomm		
F113	Swap temp1/temp2			FF19	..ipinit		
F123	Get 4 hex digits			FF24	..putass		
F130	Get hex byte			FF2A	..putas		
F154	ASCII hex to binary			FF6C	Jumbo jump table		
F15F	Input character			FFF6	Bank transfer execution		
F165	Perform [.@]	disk status		FFFA	Hard vectors		

6526 CIA 1

DB00	Inter-Processor Data							56064
DB01	X	IRQ Out	X	X	SEMAPH		Busy	56065
DB02	Data Direction Register For DB00							56066
DB02	Data Direction Register For DB01							56067
	Unused							
DB0D			IP Flag					56077
DB0E	Unused							56078
DB0F	Unused							56079

6526 CIA 2

DC00	IEEE Data In/Out							56320
DC01	User Port							56321
DC02	Data Direction Register For DC00							56322
DC02	Data Direction Register For DC01							56323
	Unused							
DC06	Timer B						L	56326
DC07							H	56327
DC08	Time Of Day Clock (TOD)						1/10 Sec.	56328
DC09							Sec.	56329
DC0A							Min.	56330
DC0B							Hour	56331
DC0C	Unused							56332
DC0D					Alarm			56333
DC0E	Unused							56334
DC0F	TOD		Timer Force				Timer Start	56335

6551 ACIA

DD00	Data Register							
DD01	IRQ	DSR	DCD	Ready Tx	Rx	OV	Error FR	PA	56577
DD02	XTRR Stop	# of Bits		Clk	Speed				56578
DD03	Parity			Echo	IRQ Tx	Rx	DTR		56579

6545 CRT Controller

D800 55296	D801 55297	Typical Value (Decimal)
0	Horizontal Total	108 or 126 or 127
1	Horizontal Char Displayed	80
2	Horizontal Sync Position	83 or 98 or 96
3	Sync Width V / H	15 or 10
4	Vertical Total	25 or 31 or 38
5	Vert Total Adjust	3 or 6 or 1
6	Vertical Displayed	25
7	Vert. Sync Position	25 or 28 or 30
8	Mode	0
9	Scan Lines	13 or 7
10	Cursor Start	96 (blink) or 0 or 6 (underline)
11	Cursor End	13 or 7
12	Display Address H	0
13	Display Address L	0
14	Cursor Address H	Varies
15	Cursor Address L	Varies
16	Light Pen In H	0
17	Light Pen In L	0

Most Register are Write Only 14/15 are Read/Write
16/17 are Read Only
Registers 10, 14 and 15 change as the cursor moves

6581 SID

DA01	Voice 1 Frequency High							55809
DA04			Saw Tooth		Ring Mod		Key	55812
DA05	Attack				Decay			55813
DA06	Sustain				Release			55814
DA0F	Voice 3 Modulating Freq Hi							55823
DA18					Volume			55832

VIC 20 Super Expander (1211A) Memory Map

Chuan Chee, St. Catharines, Ont.

Compiled 1982/03/29
Originally published in COMPUTE!

General Input/Output Routines

A000 – A001	**Vector:** RESET ($A044)
A002 – A003	**Vector:** NMI ($A077)
A004 – A008	ROM identification ('a0CBM')
A009 – A010	**Table:** function key numbers
A011 – A043	**Table:** initial function key definitions
A044 – A076	RESET routine
A077 – A08A	NMI routine
A08B – A0BE	Parse KEY (get parameters and check syntax)
A0BF – A131	Display all function key definitions
A110 – A11C	Print '" + chr$(34)' and an optional '+'
A11D – A131	Print '" + chr$(13)' and an optional '+'
A132 – A135	**Table:** ASCII string for output ('key ' backwards)
A136 – A13F	**Table:** ASCII string for output (' " + chr$(13)' backwards)
A140 – A149	**Table:** ASCII string for output (' " + chr$(34)' backwards)
A14A – A17A	Delete current function key string (key number in .X)
A17B – A1B0	Insert string into function key definition area
A1B1 – A1BE	Locate function key definition (key # in .X, return index in .Y)
A1BF – A213	**Table:** new BASIC keywords in ASCII form
A214 – A237	**Table:** vectors corresponding to new BASIC tokens ($CC to $DD)
A238 – A2A1	Initialize kernal vectors, I/O, RAM
A2A2 – A2C1	**Table:** kernal vectors (L,H)
A2C2 – A2C7	Warm start routine
A2C8 – A317	Output a character to device 3 (char in .A)
A318 – A336	End music mode
A337 – A365	Interpret keyboard matrix input
A366 – A369	**Table:** keyboard matrix code for function keys
A36A – A371	**Table:** conversion pattern for function keys
A372 – A394	IRQ routine
A395 – A3A5	Input a char from any device (device number in $99)
A3A6 – A3B3	Output a char to any device (char in .A, device num in $9A)
A3B4 – A3F1	Input each char from keyboard buffer
A3B4 – A3E7	Handle 'RUN' key
A3E8 – A3F1	Handle 'RETURN' key
A3F2 – A3FC	Input from device 0
A3FD – A406	Print an error message in GRAPHIC 0 mode (error token in .A)
A407 – A4B9	Lexically analyse BASIC source line (translate to tokens)
A4BA – A503	Print BASIC tokens in ASCII form
A504 – A529	Start new BASIC statement
A515 – A523	Handle new tokens ($CC to $D6)
A52A – A58A	Get and evaluate an expression
A558 – A58A	Handle new function tokens ($D7 to $DD)
A58B – A596	**Table:** BASIC vectors for RAM
A597 – A5A4	Change BASIC vectors during RESET

Music Routines

A5A5 – A5D0	Save current sound table (address of table in .X,.Y)
A5D1 – A601	IRQ music driver
A602 – A625	**Table:** conversion for note index to frequency
A626 – A6E5	Interpret music mode characters (char in .A)
A629 – A643	Execute 'O' command, default 3
A644 – A65D	Execute 'T' command, default 0
A65E – A674	Execute 'S' command, default 4
A675 – A686	Execute 'V' command, default 7
A687 – A693	Execute 'R' command
A694 – A69B	Execute 'P' command
A69C – A6A7	Execute 'Q' command
A6A8 –	Play new note (note index in .Y)
A6AB – A6B3	Save new sound table when previous note finishes
A6B4 – A6B9	Common return routine
A6BA – A6CD	Play notes 'A' to 'G'
A6CE – A6DA	Execute '#' command
A6DB – A6E5	Execute '$' command
A6E6 – A6EC	**Table:** conversion for notes to note index
A6ED – A6EF	**Table:** conversion for octave to base note index
A6F0 – A6F9	**Table:** conversion for tempo to duration (jiffies)

Parsing New Command Routines

A6FA –	Look for and evaluate first 1-byte and two 2-byte parameters
A6FD –	Look for and evaluate two 2-byte parameters
A700 –	Look for and evaluate one 2-byte parameter
A714 – A71B	Save one 1-byte parameter (parameter in .A, index in .Y)
A71C –	Look for and evaluate two 1-byte parameters
A71F – A72B	Look for and evaluate one 1-byte parameter
A72C – A73F	Parse GRAPHIC (get parameters and check syntax)
A740 – A762	Parse CIRCLE
A763 – A7A4	Parse DRAW
A7A5 – A7BC	Parse POINT
A7BD –	Parse COLOR
A7C8 – A7CE	Go to execute commands after parsing
A7CF – A7D8	Parse REGION
A7D9 – A7DC	Parse SCNCLR
A7DD – A7E9	Parse SOUND
A7EA – A809	Parse CHAR
A80A – A810	Parse PAINT
A811 – A817	Parse RPOT
A818 – A81B	Parse RPEN
A81C – A81F	Parse RSND
A820 – A823	Parse RCOL
A824 – A827	Parse RGR
A828 – A842	Parse RDOT
A843 – A846	Parse RJOY
A847 – A84E	Look for first 1-byte parameter
A84F – A866	Indirect jump to execute new commands (pointer to parameter save area in .X,.Y, command index in .A)
A867 – A878	**Table:** vector to execute new commands (H)
A879 – A88A	**Table:** vector to execute new commands (L)
A88B – AA22	Execute GRAPHIC
A8AB – A94E	Handle GRAPHIC 1,2,3 if previous was 0
A8D4 – A942	Transfer BASIC program to above $2000 and execute CLR
A943 – A94E	Make screen at $1E00 and character set at $1000
A94F – A9A8	Handle GRAPHIC 4
A967 – A9AB	Transfer BASIC program down to old location and execute CLR
A9AC – A9B7	Handle GRAPHIC 0 if previous was 1,2,3
A9B8 – AA22	Set up proper GRAPHIC screen
AA23 – AA28	Execute RGR
AA29 – AA6A	Execute COLOR
AA68 – AA84	Execute REGION
AA85 – AA8B	Execute RCOL
AA8C – AAE6	Execute RDOT
AAE7 – AAF1	Execute POINT
AAF2 – AB12	Execute SCNCLR
AB13 – AB22	Execute DRAW (C TO X,Y ...)
AB23 – AB34	Execute DRAW (C,X,Y TO X,Y ...)
AB35 – AB54	Execute SOUND
AB55 – AB69	Execute RSND
AB6A – AB76	Execute RPOT
AB77 – AB7D	Execute RPEN
AB7E –	Plot a single point from parameter save area
AB86 – ABE4	Plot a single point from beginning scaled X,Y coordinates
ABE5 –	Set up pointers to char and colour memory
ABFA – AC0A	Set up pointer to colour memory
AC0B –	Draw a line with a new starting coordinate
AC11 – AC92	Draw a line starting from previous coordinate (using a version of Bresenham's DDA algorithm)
AC93 – AD12	Execute CIRCLE using principal of digital differential analyser (DDA)
AD13 –	Convert starting angle to radians
AD19 – AD22	Divide FAC#1 by 16
AD23 –	Calculate new scaled X and Y coordinate on locus
AD39 – AD6B	Calculate unit offset times scaled radius
AD6C – ADDE	Execute PAINT
ADDF –	Check for possible new lower bound pivot coordinate
ADE8 – AE01	Save pivot coordinate
AE02 – AE0B	Check for possible new upper bound pivot coordinate
AE0C –	Check if able to PAINT a coordinate
AE0F – AE1E	Check if able to PAINT a coordinate (X,Y in .A,.Y)
AE1F – AE23	Move beginning scaled X,Y coordinate to .A,.Y
AE24 – AE3B	Check if coordinate has been already plotted on
AE3C – AE44	Move beginning scaled X coordinate to the right
AE45 – AE51	Move beginning scaled X coordinate 2 to the left
AE52 – AE56	Flag 'FORMULA TOO COMPLEX' error message
AE57 – AED9	Execute CHAR
AEDA – AF13	Execute RJOY
AF14 – AF33	Set up correct VIC chip screen registers
AF34 –	Save number of coordinates and colour register
AF39 – AF3E	Save colour register
AF3F – AF47	Copy beginning from ending scaled X,Y coordinate
AF48 – AF75	Scale X and Y coordinates
AF76 – AFB0	Scale X and Y coordinates to the range 0 to 159 (.X = .A*coordinate*2/256) (number of columns or rows in .A)

AFB1	– AFBA	**Table:** vector to map Y coordinate to colour memory (L)
AFBB	– AFCE	**Table:** vector to map X coordinate to character memory (L)
AFCF	– AFE2	**Table:** vector to map X coordinate to character memory (H)
AFE3	– AFE5	**Table:** bit set for colour memory
AFE6	– AFE6	(not used – contains $00)
AFE7	– AFEE	**Table:** bit mask for highres mode
AFEF	– AFF6	**Table:** bit mask for multicolour mode
AFF7	– AFFA	**Table:** bytes to plot in multicolour mode
AFFB	– AFFE	**Table:** conversion factor for VIC chip screen registers
AFFF	– AFFF	(not used – contains $AA)

General RAM Area

0024		Number of coordinates
0024		**Flag:** colour register mode ($FF = multicolour, $00 = highres)
0024	– 0025	**Pointer:** new start of variables / start of BASIC memory
0026		Temp area for building VIC chip registers / for building character byte / for saving start of BASIC (L)

Current Coordinates

0062		Ending scaled X coordinate (0 to 159)
0063		Beginning scaled X coordinate (0 to 159)
0064		Scaled X difference (absolute value)
0065		Ending scale Y coordinate (0 to 159)
0066		Beginning scaled Y coordinate (0 to 159)
0067		Scaled Y difference (absolute value–1)

For Scaling Coordinates

0069		Multiplicand–1
006A		16–bit product
006B	– 006C	10–bit multiplier

For DRAW

0067		Scaled X unit direction–1
006A		Scaled Y unit direction
006B	– 006C	Number of scaled Y units left to plot before next scaled X unit (count up)
006D	– 006E	Number of points left to plot (count up)

For PAINT

0069		**Index:** pivot coordinates save area

For CHAR

0069		Current row (0 to 19)
006A		Current column (0 to 19)
006B		Length of string
006C	– 006D	**Pointer:** string location

General RAM Area

009B		**Index:** beginning of current function key definition
009B	– 009C	**Pointer:** current char set address / byte in char set / position in screen memory / destination of byte of BASIC program to transfer
009D		**Index:** end of function key definition area
009E		Current function key number / length of current function key string
009F		Length of current function key string (count down)
009E	– 009F	**Pointer:** byte in colour memory
00AC	– 00AD	**Pointer:** current byte (function key definition, tape, scrolling)
00C3		**Flag:** 0 = have transferred BASIC program to a new location
00C3	– 00C4	**Pointer:** kernal set up / current music table / parameter save area ($003C)
00FB	– 00FC	**Pointer:** top of BASIC memory (usually same as $0284–$0285)

For KEY

028F	– 0290	**Vector:** interpret keyboard input ($A337)
02A1		Number of bytes taken by Super Expander in high memory ($88)
02A2		Number of characters in function key definition
02A3		**Index:** current byte of function key string
02A4		Length of function key string (amount left to output)

For Music

02A5		Previous character in music mode
02A6		Music mode ($80 = on)
02A7		Screen echo ($50 = on, $00 = off)
02A8		Current voice (sound register–1)
02A9		Current note index
02AA		Current duration (jiffies)
02AB		Current sound amplitude (volume * 2)
02AC		Current octave (base note index)
02AD		Voice 1 note index (+$80)
02AE		Voice 1 duration count down (jiffies)
02AF		Voice 2 note index (+$80)
02B0		Voice 2 duration count down (jiffies)
02B1		Voice 3 note index (+$80)
02B2		Voice 3 duration count down (jiffies)
02B3		Voice 4 note index (+$80)
02B4		Voice 4 duration count down (jiffies)
02B5	– 02BF	(for expansion)

For Execution Of New Commands

02C0	– 02C2	**Jump table:** execute new commands (JMP $A84F)
02C3		Current VIC chip left margin register
02C4		Current VIC chip top margin register
02C5		Current VIC chip number of columns register
02C6		Current VIC chip number of rows register
02C7		Current row of cursor
02C8		Current GRAPHIC mode
02C9		(for expansion)
02CA		Current colour register parameter (while plotting)
02CB		Current screen colour
02CC		Current border colour
02CD		Current character colour
02CE		Current auxiliary colour
02CF		**Index:** parameter save area (while plotting)
02D0		Current character set address page
02D1		Usual character set address page ($80)
02D2	– 02D3	**Pointer:** old limit of BASIC memory
02D4		Old screen memory page
02D5		Last scaled X coordinate (0 to 159)
02D6		Last scaled Y coordinate (0 to 159)
02D7		**Flag:** $00 = DRAW C,X,Y TO, $01 = DRAW C TO / current number of out of range coordinates ($00 = within range)
02D8		Old number of out of range coordinates ($00 = within range)
02D9		**Index:** parameter save area (while getting parameters)
02DA	– 02FF	(for expansion)

Operating System Vectors

0300	– 0301	**Vector:** error message	($A3FD)
0302	– 0303	**Vector:** BASIC warm start	($C483)
0304	– 0305	**Vector:** lexically analyse BASIC source line	($A407)
0306	– 0307	**Vector:** print BASIC tokens in ASCII form	($A4BA)
0308	– 0309	**Vector:** start new BASIC statement	($A504)
030A	– 030B	**Vector:** get and evaluate an expression	($A52A)
0314	– 0315	**Vector:** IRQ	($A372)
0316	– 0317	**Vector:** BRK instruction	($A2C2)
0318	– 0319	**Vector:** NMI	($FEAD)
031A	– 031B	**Vector:** BASIC OPEN statement	($F40A)
031C	– 031D	**Vector:** BASIC CLOSE statement	($F34A)
031E	– 031F	**Vector:** set input	($F2C7)
0320	– 0321	**Vector:** set output	($F309)
0322	– 0323	**Vector:** restore I/O	($F3F3)
0324	– 0325	**Vector:** input a character	($A395)
0326	– 0327	**Vector:** output a character	($A3A6)
0328	– 0329	**Vector:** test 'STOP' key	($F770)
032A	– 032B	**Vector:** BASIC GET statement	($F1F5)
032C	– 032D	**Vector:** abort I/O	($F3EF)
032E	– 032F	**Vector:** user BRK instruction	($A2C2)
0330	– 0331	**Vector:** BASIC LOAD statement	($F549)
0332	– 0333	**Vector:** BASIC SAVE statement	($F685)
0334	– 033B	(for expansion)	

Save Area

033C	– 03F8	Save area: parameter passing / pivot coordinates (PAINT)

For CIRCLE

033C		**Index:** X or Y
0347	– 0348	Old scaled X coordinate on locus
0349	– 034A	Old scaled Y coordinate on locus
034B	– 034C	New scaled X coordinate on locus
034D	– 034E	New scaled Y coordinate on locus
034F	– 0353	Floating point unit offset X coordinate
0355	– 0359	Floating point unit offset Y coordinate

Instruction Set Summary

Instr	Addressing Mode	Assembler Format	Operation	Op Code Hex	Op Code Dec	Bytes	Clock Cycles	Status Register - P (N V D I Z C)	Instr
ADC	Immediate	ADC #oper	.A + # + C → .A, C	69	105	2	2	✓ ✓ – – ✓ ✓	**ADC**
	Zero Page	ADC addr	.A + [addr] + C → .A, C	65	101	2	3		
	Zero Page, X	ADC addr, X	.A + [addr + .X] + C → .A, C	75	117	2	4		
	Absolute	ADC ADDR	.A + [ADDR] + C → .A, C	6D	109	3	4		
	Absolute, X	ADC ADDR, X	.A + [ADDR + .X] + C → .A, C	7D	125	3	4*		
	Absolute, Y	ADC ADDR, Y	.A + [ADDR + .Y] + C → .A, C	79	121	3	4*		
	(Indirect, X)	ADC (addr, X)	.A + [[addr + .X + 1, addr + .X]] + C → .A, C	61	97	2	6		
	(Indirect),Y	ADC (addr),Y	.A + [[addr + 1, addr] + .Y] + C → .A, C	71	113	2	5*		
AND	Immediate	AND #oper	.A ∩ # → .A	29	41	2	2	✓ – – – ✓ –	**AND**
	Zero Page	AND addr	.A ∩ [addr] → .A	25	37	2	3		
	Zero Page, X	AND addr, X	.A ∩ [addr + .X] → .A	35	53	2	4		
	Absolute	AND ADDR	.A ∩ [ADDR] → .A	2D	45	3	4		
	Absolute, X	AND ADDR, X	.A ∩ [ADDR + .X] → .A	3D	61	3	4*		
	Absolute, Y	AND ADDR, Y	.A ∩ [ADDR + .Y] → .A	39	57	3	4*		
	(Indirect, X)	AND (addr, X)	.A ∩ [[addr + .X + 1, addr + .X]] → .A	21	33	2	6		
	(Indirect),Y	AND (addr),Y	.A ∩ [[addr + 1, addr] + .Y] → .A	31	49	2	5*		
ASL	Accumulator	ASL A	.A (←) → .A ; 0 → bit 0, bit7 → C	0A	10	1	2	✓ – – – ✓ ✓	**ASL**
	Zero Page	ASL addr	[addr] (←) → [addr] "	06	6	2	5		
	Zero Page, X	ASL addr, X	[addr + .X] (←) → [addr + .X] "	16	22	2	6		
	Absolute	ASL ADDR	[ADDR] (←) → [ADDR] "	0E	14	3	6		
	Absolute, X	ASL ADDR, X	[ADDR + .X] (←) → [ADDR + .X] "	1E	30	3	7		
BCC	Relative	BCC oper	Branch on C = 0	90	144	2	2*	– – – – – –	**BCC**
BCS	Relative	BCS oper	Branch on C = 1	B0	176	2	2*	All Branches * - Add 1 if branch to same page * - Add 2 if branch to diff page	**BCS**
BEQ	Relative	BEQ oper	Branch on Z = 1	F0	240	2	2*		**BEQ**
BNE	Relative	BNE oper	Branch on Z = 0	D0	208	2	2*		**BNE**
BMI	Relative	BMI oper	Branch on N = 1	30	48	2	2*		**BMI**
BPL	Relative	BPL oper	Branch on N = 0	10	16	2	2*		**BPL**
BVS	Relative	BVS oper	Branch on V = 1	70	112	2	2*		**BVS**
BVC	Relative	BVC oper	Branch on V = 0	50	80	2	2*		**BVC**
BIT	Zero Page	BIT addr	.A ∩ [addr] ; bit7 → N, bit6 → V	24	36	2	3	N V D I Z C	**BIT**
	Absolute	BIT ADDR	.A ∩ [ADDR] "	2C	44	3	4	b_7 b_6 – – ✓ –	
BRK	Implied	BRK 1→B flag	PC + 2↓P↓, [FFFE]→PCL, [FFFF]→PCH	00	0	1	7	– – – 1 – –	**BRK**
CLC	Implied	CLC	0→C	18	24	1	2	– – – – – 0	**CLC**
CLD	Implied	CLD	0→D	D8	216	1	2	– – 0 – – –	**CLD**
CLI	Implied	CLI	0→I	58	88	1	2	– – – 0 – –	**CLI**
CLV	Implied	CLV	0→V	B8	184	1	2	– 0 – – – –	**CLV**
CMP	Immediate	CMP #oper	.A – #	C9	201	2	2	✓ – – – ✓ ✓	**CMP**
	Zero Page	CMP addr	.A – [addr]	C5	197	2	3		
	Zero Page, X	CMP addr, X	.A – [addr + .X]	D5	213	2	4		
	Absolute	CMP ADDR	.A – [ADDR]	CD	205	3	4		
	Absolute, X	CMP ADDR, X	.A – [ADDR + .X]	DD	221	3	4*		
	Absolute, Y	CMP ADDR, Y	.A – [ADDR + .Y]	D9	217	3	4*		
	(Indirect, X)	CMP (addr, X)	.A – [[addr + .X + 1, addr + .X]]	C1	193	2	6		
	(Indirect),Y	CMP (addr),Y	.A – [[addr + 1, addr] + .Y]	D1	209	2	5*		
CPX	Immediate	CPX #oper	.X – #	E0	224	2	2	✓ – – – ✓ ✓	**CPX**
	Zero Page	CPX addr	.X – [addr]	E4	228	2	3		
	Absolute	CPX ADDR	.X – [ADDR]	EC	236	3	4		
CPY	Immediate	CPY #oper	.Y – #	C0	192	2	2	✓ – – – ✓ ✓	**CPY**
	Zero Page	CPY addr	.Y – [addr]	C4	196	2	3		
	Absolute	CPY ADDR	.Y – [ADDR]	CC	204	3	4		
DEC	Zero Page	DEC addr	[addr] – 1 → [addr]	C6	198	2	5	✓ – – – ✓ –	**DEC**
	Zero Page, X	DEC addr, X	[addr + .X] – 1 → [addr + .X]	D6	214	2	6		
	Absolute	DEC ADDR	[ADDR] – 1 → [ADDR]	CE	206	3	6		
	Absolute, X	DEC ADDR, X	[ADDR + .X] – 1 → [ADDR + .X]	DE	222	3	7		
DEX	Implied	DEX	.X – 1 → .X	CA	202	1	2	✓ – – – ✓ –	**DEX**
DEY	Implied	DEY	.Y – 1 → .Y	88	136	1	2		**DEY**
EOR	Immediate	EOR #oper	.A ⊎ # → .A	49	73	2	2	✓ – – – ✓ –	**EOR**
	Zero Page	EOR addr	.A ⊎ [addr] → .A	45	69	2	3		
	Zero Page, X	EOR addr, X	.A ⊎ [addr + .X] → .A	55	85	2	4		
	Absolute	EOR ADDR	.A ⊎ [ADDR] → .A	4D	77	3	4		
	Absolute, X	EOR ADDR, X	.A ⊎ [ADDR + .X] → .A	5D	93	3	4*		
	Absolute, Y	EOR ADDR, Y	.A ⊎ [ADDR + .Y] → .A	59	89	3	4*		
	(Indirect, X)	EOR (addr, X)	.A ⊎ [[addr + .X + 1, addr + .X]] → .A	41	65	2	6		
	(Indirect),Y	EOR (addr),Y	.A ⊎ [[addr + 1, addr] + .Y] → .A	51	81	2	5*		
INC	Zero Page	INC addr	[addr] + 1 → [addr]	E6	230	2	5	✓ – – – ✓ –	**INC**
	Zero Page, X	INC addr, X	[addr + .X] + 1 → [addr + .X]	F6	246	2	6		
	Absolute	INC ADDR	[ADDR] + 1 → [ADDR]	EE	238	3	6		
	Absolute, X	INC ADDR, X	[ADDR + .X] + 1 → [ADDR + .X]	FE	254	3	7		
INX	Implied	INX	.X + 1 → .X	E8	232	1	2	✓ – – – ✓ –	**INX**
INY	Implied	INY	.Y + 1 → .Y	C8	200	1	2		**INY**
JMP	Absolute	JMP ADDR	[PC + 1] → PCL, [PC + 2] → PCH	4C	76	3	3	– – – – – –	**JMP**
	Indirect	JMP (ADDR)	[ADDR] → PCL, [ADDR + 1] → PCH	6C	108	3	5		
JSR	Absolute	JSR ADDR	PC + 2↓, [PC + 1] → PCL, [PC + 2] → PCH	20	32	3	6		**JSR**

Instr	Addressing Mode	Assembler Format	Operation	Op Code Hex	Op Code Dec	Bytes	Clock Cycles	Status Register - P N	V	D	I	Z	C	Instr
LDA	Immediate	LDA #oper	# → .A	A9	169	2	2							**LDA**
	Zero Page	LDA addr	[addr] → .A	A5	165	2	3	✓	–	–	–	✓	–	
	Zero Page, X	LDA addr, X	[addr + .X] → .A	B5	181	2	4							
	Absolute	LDA ADDR	[ADDR] → .A	AD	173	3	4							
	Absolute, X	LDA ADDR, X	[ADDR + .X] → .A	BD	189	3	4*							
	Absolute, Y	LDA ADDR, Y	[ADDR + .Y] → .A	B9	185	3	4*							
	(Indirect, X)	LDA (addr, X)	[[addr + .X + 1, addr + .X]] → .A	A1	161	2	6							
	(Indirect),Y	LDA (addr),Y	[[addr + 1, addr] + .Y] → .A	B1	177	2	5*							
LDX	Immediate	LDX #oper	# → .X	A2	162	2	2	N	V	D	I	Z	C	**LDX**
	Zero Page	LDX addr	[addr] → .X	A6	166	2	3	✓	–	–	–	✓	–	
	Zero Page, Y	LDX addr, Y	[addr + .Y] → .X	B6	182	2	4							
	Absolute	LDX ADDR	[ADDR] → .X	AE	174	3	4							
	Absolute, Y	LDX ADDR, Y	[ADDR + .Y] → .X	BE	190	3	4*							
LDY	Immediate	LDY #oper	# → .Y	A0	160	2	2	N	V	D	I	Z	C	**LDY**
	Zero Page	LDY addr	[addr] → .Y	A4	164	2	3	✓	–	–	–	✓	–	
	Zero Page, X	LDY addr, X	[addr + .X] → .Y	B4	180	2	4							
	Absolute	LDY ADDR	[ADDR] → .Y	AC	172	3	4							
	Absolute, X	LDY ADDR, X	[ADDR + .X] → .Y	BC	188	3	4*							
LSR	Accumulator	LSR A	.A (→) → .A ; 0→bit7, bit0→C	4A	74	1	2	N	V	D	I	Z	C	**LSR**
	Zero Page	LSR addr	[addr] (→) → [addr] "	46	70	2	5	0	–	–	–	✓	✓	
	Zero Page, X	LSR addr, Y	[addr + .X] (→) → [addr + .X] "	56	86	2	6							
	Absolute	LSR ADDR	[ADDR] (→) → [ADDR] "	4E	78	3	6							
	Absolute, X	LSR ADDR, X	[ADDR + .X] (→) → [ADDR + .X] "	5E	94	3	7							
NOP	Implied	NOP	No OPeration	EA	234	1	2	–	–	–	–	–	–	**NOP**
ORA	Immediate	ORA #oper	.A ∪ # → .A	09	9	2	2	N	V	D	I	Z	C	**ORA**
	Zero Page	ORA addr	.A ∪ [addr] → .A	05	5	2	3	✓	–	–	–	✓	–	
	Zero Page, X	ORA addr, X	.A ∪ [addr + .X] → .A	15	21	2	4							
	Absolute	ORA ADDR	.A ∪ [ADDR] → .A	0D	13	3	4							
	Absolute, X	ORA ADDR, X	.A ∪ [ADDR + .X] → .A	1D	29	3	4*							
	Absolute, Y	ORA ADDR, Y	.A ∪ [ADDR + .Y] → .A	19	25	3	4*							
	(Indirect, X)	ORA (addr, X)	.A ∪ [[addr + .X + 1, addr + .X]] → .A	01	1	2	6							
	(Indirect),Y	ORA (addr),Y	.A ∪ [[addr + 1, addr] + .Y] → .A	11	17	2	5*							
PHA	Implied	PHA	.A ↓, SP – 1 → SP	48	72	1	3	N	V	D	I	Z	C	**PHA**
PLA	Implied	PLA	.A ↑, SP + 1 → SP	68	104	1	4	–	–	–	–	–	–	**PLA**
PHP	Implied	PHP	.P ↓, SP – 1 → SP	08	8	1	3	All Push/Pulls xcpt PLP						**PHP**
PLP	Implied	PLP	.P ↑, SP + 1 → SP	28	40	1	4	from stack						**PLP**
ROL	Accumulator	ROL A	.A (←) → .A ; C→bit0, bit7→C	2A	42	1	2	N	V	D	I	Z	C	**ROL**
	Zero Page	ROL addr	[addr] (←) → [addr] "	26	38	2	5	✓	–	–	–	✓	✓	
	Zero Page, X	ROL addr, X	[addr + .X] (←) → [addr + .X] "	36	54	2	6							
	Absolute	ROL ADDR	[ADDR] (←) → [ADDR] "	2E	46	3	6							
	Absolute, X	ROL ADDR, X	[ADDR + .X] (←) → [ADDR + .X] "	3E	62	3	7							
ROR	Accumulator	ROR A	.A (→) → .A ; C→bit7, bit0→C	6A	106	1	2	N	V	D	I	Z	C	**ROR**
	Zero Page	ROR addr	[addr] (→) → [addr] "	66	102	2	5	✓	–	–	–	✓	✓	
	Zero Page, X	ROR addr, Y	[addr + .X] (→) → [addr + .X] "	76	118	2	6							
	Absolute	ROR ADDR	[ADDR] (→) → [ADDR] "	6E	110	3	6							
	Absolute, X	ROR ADDR, X	[ADDR + .X] (→) → [ADDR + .X] "	7E	126	3	7							
RTI	Implied	RTI	P ↑, PC ↑, SP + 3 → SP, PC + 1 → PC	40	64	1	6	from stack						**RTI**
RTS	Implied	RTS	PC ↑, SP + 2 → SP, PC + 1 → PC	60	96	1	6	–	–	–	–	–	–	**RTS**
SBC	Immediate	SBC #oper	.A – # – C̄ → .A, C C̄ = Borrow	E9	233	2	2	N	V	D	I	Z	C	**SBC**
	Zero Page	SBC addr	.A – [addr] – C̄ → .A, C	E5	229	2	3	✓	✓	–	–	✓	✓	
	Zero Page, X	SBC addr, X	.A – [addr + .X] – C̄ → .A, C	F5	245	2	4							
	Absolute	SBC ADDR	.A – [ADDR] – C̄ → .A, C	ED	237	3	4							
	Absolute, X	SBC ADDR, X	.A – [ADDR + .X] – C̄ → .A, C	FD	253	3	4*							
	Absolute, Y	SBC ADDR, Y	.A – [ADDR + .Y] – C̄ → .A, C	F9	249	3	4*							
	(Indirect, X)	SBC (addr, X)	.A – [[addr + .X + 1, addr + .X]] – C̄ → .A, C	E1	225	2	6							
	(Indirect),Y	SBC (addr),Y	.A – [[addr + 1, addr] + .Y] – C̄ → .A, C	F1	241	2	5*							
								N	V	D	I	Z	C	
SEC	Implied	SEC	1→C	38	56	1	2	–	–	–	–	–	1	**SEC**
SED	Implied	SED	1→D	F8	248	1	2	–	–	1	–	–	–	**SED**
SEI	Implied	SEI	1→I	78	120	1	2	–	–	–	1	–	–	**SEI**
STA	Zero Page	STA addr	.A → [addr]	85	133	2	3	N	V	D	I	Z	C	**STA**
	Zero Page, X	STA addr, X	.A → [addr + .X]	95	149	2	4	–	–	–	–	–	–	
	Absolute	STA ADDR	.A → [ADDR]	8D	141	3	4							
	Absolute, X	STA ADDR, X	.A → [ADDR + .X]	9D	157	3	5							
	Absolute, Y	STA ADDR, Y	.A → [ADDR + .Y]	99	153	3	5							
	(Indirect, X)	STA (addr, X)	.A → [[addr + .X + 1, addr + .X]]	81	129	2	6							
	(Indirect),Y	STA (addr),Y	.A → [[addr + 1, addr] + .Y]	91	145	2	6							
STX	Zero Page	STX addr	.X → [addr]	86	134	2	3	N	V	D	I	Z	C	**STX**
	Zero Page, Y	STX addr, Y	.X → [addr + .Y]	96	150	2	4	–	–	–	–	–	–	
	Absolute	STX ADDR	.X → [ADDR]	8E	142	3	4							
STY	Zero Page	STY addr	.Y → [addr]	84	132	2	3	N	V	D	I	Z	C	**STY**
	Zero Page, X	STY addr, X	.Y → [addr + .X]	94	148	2	4	–	–	–	–	–	–	
	Absolute	STY ADDR	.Y → [ADDR]	8C	140	3	4							
TAX	Implied	TAX	.A → .X	AA	170	1	2	N	V	D	I	Z	C	**TAX**
TXA	Implied	TXA	.X → .A	8A	138	1	2	✓	–	–	–	✓	–	**TXA**
TAY	Implied	TAY	.A → .Y	A8	168	1	2							**TAY**
TYA	Implied	TYA	.Y → .A	98	152	1	2	All Transfers xcpt TXS						**TYA**
TSX	Implied	TSX	SP → .X	BA	186	1	2							**TSX**
TXS	Implied	TXS	.X → SP	9A	154	1	2	–	–	–	–	–	–	**TXS**

MCS65XX Microprocessor Instruction Set Alphabetic Sequence

ADC	Add memory to accumulator with carry.
AND	AND memory with accumulator.
ASL	Shift left one bit (memory or accumulator).
BCC	Branch on carry clear.
BCS	Branch on carry set.
BEQ	Branch on result zero.
BIT	Test bits in memory with accumulator.
BMI	Branch on result minus.
BNE	Branch on result not zero.
BPL	Branch on result plus.
BRK	Force break.
BVC	Branch on overflow clear.
BVS	Branch on overflow set.
CLC	Clear carry flag.
CLD	Clear decimal mode.
CLI	Clear interrupt disable bit.
CLV	Clear overflow flag.
CMP	Compare memory and accumulator.
CPX	Compare memory and index 'X'.
CPY	Compare memory and index 'Y'.
DEC	Decrement memory by one.
DEX	Decrement index 'X' by one.
DEY	Decrement index 'Y' by one.
EOR	Exclusive-OR memory with accumulator.
INC	Increment memory by one.
INX	Increment index 'X' by one.
INY	Increment index 'Y' by one.
JMP	Jump to new location.
JSR	Jump to new location saving return address.
LDA	Load acculmulator with memory.
LDX	Load index 'X' with memory.
LDY	Load index 'Y' with memory.
LSR	Shift right one bit (memory or accumulator).
NOP	No operation.
ORA	OR memory with accumulator.
PHA	Push accumulator on stack.
PHP	Push processor status on stack.
PLA	Pull accumulator from stack.
PLP	Pull processor status from stack.
ROL	Rotate one bit left (memory or accumulator).
ROR	Rotate one bit right (memory or accumulator).
RTI	Return from interrupt.
RTS	Return from subroutine.
SBC	Subtract memory from accumulator with borrow.
SEC	Set carry flag.
SED	Set decimal mode.
SEI	Set interrupt disable status.
STA	Store accumulator in memory.
STX	Store index 'X' in memory.
STY	Store index 'Y' in memory.
TAX	Transfer accumulator to index 'X'.
TAY	Transfer accumulator to index 'Y'.
TSX	Transfer stack pointer to index 'X'.
TXA	Transfer index 'X' to accumulator.
TXS	Transfer index 'X' to stack pointer.
TYA	Transfer index 'Y' to accumulator.

Addressing Modes

Accumulator Addressing - This form of addressing is represented with a one byte instruction, implying an operation on the accumulator.

Immediate Addressing - In immediate addressing, the operand is contained in the second byte of the instruction, with no further memory addressing required.

Absolute Addressing - In absolute addressing, the second byte of the instruction specifies the eight low order bits of the effective address while the third byte specifies the eight high order bits. Thus, the absolute addressing mode allows access to the entire 65k bytes of addressable memory.

Zero Page Addressing - The zero page instructions allow for shorter code and execution times by only fetching the second byte of the instructions and assuming a zero high address byte. Careful use of the zero page can result in significant increase in code efficiency.

Indexed Zero Page Addressing - (X, Y Indexing) - This form of addressing is used in conjunction with the index register and is referred to as "Zero Page, X" or "Zero Page, Y". The effective address is calculated by adding the second byte to the contents of the index register. Since this is a form of "Zero Page" addressing, the content of the second byte references a location in page zero. Additionally due to the "Zero Page" addressing nature of this mode, no carry is added to the high order 8 bits of memory and crossing of page boundaries does not occur.

Indexed Absolute Addressing - (X, Y Indexing) - This form of addressing is used in conjunction with X and Y index register and is referred to as "Absolute, X", and "Absolute, Y". The effective address is formed by adding the contents of X or Y to the address contained in the second and third bytes on the instruction. This mode allows the index register to contain the index or count value and the instruction to contain the base address. This type of indexing allows any location referencing and the index to modify multiple fields resulting in reduced coding and execution time.

Implied Addressing - In the implied addressing mode, the address containing the operand is implicitly stated in the operation code of the instruction.

Relative Addressing - Relative addressing is used only with branch instructions and establishes a destination for the conditional branch.
the second byte of the instruction becomes the operand which is an "offset" added to the contents of the lower eight bits of the program counter when the counter is set at the next instruction. The range of the offset is -128 to +127 bytes from the next instruction.

Indexed Indirect Addressing - In indexed indirect addressing (referred to as (Indirect, X)), the second byte of the instruction is added to the contents of the X index register, discarding the carry. The result of the addition points to a memory location on page zero whose contents is the low order eights bits of the effective address. The next memory location in page zero contains the high order eight bits of the effective address. Both memory locations specifying the high and low order bytes of the effective address must be in page zero.

Indirect Indexed Addressing - In indirect indexed addressing (referred to as (Indirect), Y), the second byte of the instruction points to a memory location in page zero. The contents of this memory location is added to the contents of the Y register, the result being the low order eight bits of the effective address. The carry from this addition is added to the contents of the next page zero memory location, the result being the high order eight bits of the effective address.

Absolute Indirect - The second byte of the instruction contains the low order eight bits of a memory location. The high order eight bits of that memory location is contained in the third byte of the instruction. The contents of the fully specified memory location is the low order byte of the effective address which is loaded into the sixteen bits of the program counter.

BASIC 4.0 / 2.0 Kernal Routines

CBM Label	Address Hex	Address Dec	Operation	Registers In .A	Registers In .X	Registers In .Y	Registers Out .A	Registers Out .X	Registers Out .Y
CHKIN	FFC6	65478	Open channel for input		LF#		alt.		
CHKOUT	FFC9	65481	Open channel for output		LF#		alt.		
CHRIN	FFCF	65487	Input character from channel				data	alt.	
CHROUT	FFD2	65490	Output character to channel	data					
CLALL	FFE7	65511	Close all channels and files				alt.	alt.	
CLOSE	FFC3	65475	Close a specified logical file	LF#			alt.	alt.	alt.
CLRCHN	FFCC	65484	Restore default I/O devices				alt.	alt.	
CSYS	FFDE	65502	SYS vector		addr lo	addr hi	alt.	alt.	alt.
CVERF	FFDB	65499	Verify ram from a device		start lo	start hi		end lo + 1	end hi
GETIN	FFE4	65508	Get character from current input device				data	alt.	alt.
LOAD	FFD5	65493	Load ram from a device		start lo	start hi		end lo + 1	end hi
OPEN	FFC0	65472	Open a logical file				alt.	alt.	alt.
SAVE	FFD8	65496	Save 'ram' to device	txttab lo	start lo	start hi		end lo + 1	end hi
STOP	FFE1	65505	Scan stop key depressed	yes: .Z = 1, no .A = last row kybd scan					
UDTIM	FFEA	65514	Increment real time clock				alt.	alt.	

alt. = altered

VIC 20 And Commodore 64 Kernal Routines

CBM Label	Address Hex	Address Dec	Operation	Registers In .A	Registers In .X	Registers In .Y	Registers Out .A	Registers Out .X	Registers Out .Y
ACPTR	FFA5	65445	Input byte from Serial Port				data	alt.	
CHKIN	FFC6	65478	Open channel for input		LF#		alt.		
CHKOUT	FFC9	65481	Open channel for output		LF#		alt.		
CHRIN	FFCF	65487	Input character from channel				data	alt.	
CHROUT	FFD2	65490	Output character to channel	data					
CIOUT	FFA8	65448	Output byte to serial port	data					
CINT	FF81	65409	Initialize screen editor				alt.	alt.	alt.
CLALL	FFE7	65511	Close all channels and files				alt.	alt.	
CLOSE	FFC3	65475	Close a specified logical file	LF#			alt.	alt.	alt.
CLRCHN	FFCC	65484	Restore default I/O devices				alt.	alt.	
GETIN	FFE4	65508	Get character from current input device				data	alt.	alt.
IOBASE	FFF3	65523	Returns base address of I/O devices					addr lo	addr hi
IOINIT	FF84	65412	Initialize Input/Output				alt.	alt.	alt.
LISTEN	FFB1	65457	Command devices on the serial bus to listen	DEV#					
LOAD	FFD5	65493	Load (.A = 0) or Verify (.A = 1) 'ram' from a device		start lo	start hi		end lo + 1	end hi
MEMBOT	FF9C	65436	Read (.C = 1) or Set (.C = 0) the bottom of memory	.C = 0:	bot lo	bot hi	.C = 1:	bot lo	bot hi
MEMTOP	FF99	65433	Read (.C = 1) or Set (.C = 0) the top of memory	.C = 0:	top lo	top hi	.C = 1:	top lo	top hi
OPEN	FFC0	65472	Open a logical file				alt.	alt.	alt.
PLOT	FFF0	65520	Read (.C = 1) or Set (.C = 0) x, y cursor position		row	col		row	col
RAMTAS	FF87	65415	Init. ram, allocate tape buff, set screen $0400				alt.	alt.	alt.
RDTIM	FFDE	65502	Read real time clock				msb	msb2	lsb
READST	FFB7	65463	Read I/O status word				ST		
RESTOR	FF8A	65418	Restore default I/O vectors				alt.	alt.	alt.
SAVE	FFD8	65496	Save 'ram' to device	txttab lo	start lo	start hi		end lo + 1	end hi
SCNKEY	FF9F	65439	Scan keyboard				alt.	alt.	alt.
SCREEN	FFED	65517	Return screen size in rows & columns					#rows	#cols
SECOND	FF93	65427	Send secondary address after 'listen'	SA OR $60					
SETLFS	FFBA	65466	Set logical, first, and second addresses	LF#	DEV#	SA			
SETMSG	FF90	65424	Enable/Disable 'Kernal' messages	.A val: $40 control msgs on. $80 error msgs on. $00 off					
SETNAM	FFBD	65469	Set file name	len	addr lo	addr hi			
SETTIM	FFDB	65499	Set real time clock	msb	msb2	lsb			
SETTMO	FFA2	65442	Set (.A<#128) Reset (.A>#127) IEEE timeout						
STOP	FFE1	65505	Scan stop key depressed	yes: .Z = 1, no .A = last row kybd scan					
TALK	FFB4	65460	Command serial bus device to 'talk'	DEV#					
TKSA	FF96	65430	Send secondary address after 'talk'	SA					
UDTIM	FFEA	65514	Increment real time clock				alt.	alt.	
UNLSN	FFAE	65454	Command serial bus to 'unlisten'				alt.		
UNTLK	FFAB	65451	Command serial bus to 'untalk'				alt.		
VECTOR	FF8D	65421	Store (.C = 1) or Restore (.C = 0) ram vectors	.C = 1:	tabl lo	tabl hi	.C = 0:	tabl lo	tabl hi

alt. = altered

User Callable ROM Subroutines

Some I/O routines require extra memory set up. See the appropriate Memory Map. Data within parenthesis are for Basic 2.0/4.0 users. (Direct call) indicates no required set up.

#	Entry Point For: 2.0		4.0		VIC 20		C64		Operation	Registers In .A	.X	.Y	Registers Out .A	.X	.Y
1	C2D8	49880	B350	45904	C3BB	50107	A3BB	41915	Open Up Space In BASIC Text New:	AryTop Lo		AryTop Hi	Unaltered		
2	C328	49960	B3A0	45984	C408	50184	A408	41992	Check Available Memory (called by 1)	(same as above) Start address of move in $5F, 60 (5C, 5D)					
3	C355	50005	B3CD	46029	C435	50229	A435	42037	?OUT OF MEMORY	(direct call)					
4	C357	50007	BC3F	48191	C437	50231	A437	42039	Send BASIC Error Message	Error #					
5	C389	50057	B3FF	46079	C474	50292	A474	42100	Warm start, BASIC	(direct call)					
6	C399	49960	B40D	46093	C48A	50314	A48A	42122	Main CHRGET entry	(direct call) $7A = #$FF, $7B = #$01 ($77, 78);01FF = Basic Inbuf-1					
7	C3AB	50091	B41F	46111	C49C	50220	A49C	42028	Crunch tokens, insert line		Inbuf len.				
8	C439	50233	B4AD	46253	C52A	50474	A52A	42282	Fix chaining, CLR, & READY.	(direct call)					
9	C442	50242	B4B6	46262	C533	50483	A533	42291	Fix chaining	(direct call)					
10	C46F	50287	B4E2	46306	C560	50528	A560	42336	Receive line from keyboard	(direct call) $7A = #$FF, $7B = #$01					
11	C495	50213	B4FB	46331	C579	50553	A579	42361	Crunch tokens (called by 7)	.X = Inbuf Len. ($0200,X) = #$00					
12	C52C	50476	B5A3	46499	C613	50707	A613	42515	Find line in BASIC	StrtBAS Lo	StrtBAS Hi				
13	C55D	50525	B5D4	46548	C642	50754	A642	42562	Do NEW	(direct call)					
14	C572	50546	B5E9	46569	C659	50777	A659	42585	Reset BASIC and do CLR	(direct call)					
15	C575	50549	B5EC	46572	C65E	50782	A65E	42590	Do CLR	(direct call)					
16	C5A7	50599	B622	46626	C68E	50830	A68E	42638	Reset Chrget to Start of BASIC	(direct call)			StrtBAS Hi		
17	C6C4	50884	B74A	46922	C857	51287	A857	43095	Continue BASIC execution [CONT]	CurLin Lo		CurLin Hi			
18	C873	51315	B8F6	47350	C96B	49771	A96B	41579	Get fixed-pt number from BASIC text	Address of text in Chrget ptr; $7A, 7B ($77, 78)					
19	C9DE	49686	BADB	47835	CAD3	51923	AAD3	43731	Send RETURN, LF if in screen mode	(direct call)			LF (#$0A)		
20	C9E2	49690	BADF	47839	CAD7	51927	AAD7	43735	Send RETURN, LINEFEED	(direct call)			LF (#$0A)		
21	CA1C	51740	BB1D	47901	CB1E	51998	AB1E	43806	Print string from A, Y	Addr Lo		Addr Hi			
22	CA22	51746	BB23	47907	CB24	52004	AB24	43812	Print pre-computed string	Length	Addr in $22,23 ($1F,20)				
23	CA43	51779	BB44	47940	CB45	52037	AB45	43845	Print '?'	(direct call)					
24	CA45	51781	BB46	47942	CB47	52039	AB47	43847	Print char (output .A to device)	Char			Char		
25	CC9F	52383	BD98	48536	CD9E	52638	AD9E	44446	Evaluate Result string $0D = #$FF	Address of Expression			Addr Lo		Addr Hi
									Expression ($07) numeric $0D = #$00	In Chrget Pointer			result in Acc#1		
26	CDF8	52728	BEF5	48885	CEFF	52991	AEFD	44797	Check for comma	(direct call)			Char		
27	CDF7	52727	BEF2	48882	CEFA	52986	AEFA	44794	Check for '('	(direct call)			Char		
28	CDF4	52724	BEEF	48879	CEF7	52983	AEF7	44791	Check for ')'	(direct call)			Char		
29	CE03	52739	BF00	48896	CF08	53000	AF08	44808	Send 'SYNTAX ERROR'	(direct call)					
30	CFC9	53193	C187	49543	D0E7	53479	B0E7	45287	Find fl-pt variable, given name				VarAddr Lo		VarAddr Hi
31	D069	53353	C2B9	49849	D185	53637	B185	45445	Bump Variable Addr by 2 (called by 30)	Name in $45, 46 ($42, 43)			VarAddr Lo		VarAddr Hi
32	D09A	53290	C2EA	49898	D1BF	53695	B1BF	45503	Float to Fixed conversion in Acc#1	(direct call)					
33	D26D	53869	C4BC	50364	D391	54049	B391	45857	Fixed to Float conversion in Acc#1	(direct call)					
34	D67B	54907	C8D7	51415	D79E	55086	B79E	46894	Get Acc#1 least significant byte to X register					Data	
35	D68F	54927	C8EB	51435	D7B5	55221	B7B5	47029	Evaluate string [VAL]	Address – (Chrget Ptr.)			Fl. Pt. result in Acc#1		
36	D69D	54931	C8EF	51439	D7B9	55225	B7B9	47033	Evaluate string from X, Y (above + 4)		Addr Lo	Addr Hi	Fl. Pt. result in Acc#1		
37	D6C6	54982	C921	49697	D7EB	55275	B7EB	47083	Get two params for POKE, WAIT	Address = (Chrget Ptr.)			.X = Pram2, Pram1 in Acc#1 (fxd pt)		
38	D773	55155	C99D	49709	D867	55399	B867	47207	Add (from memory)	Addr Lo		Addr Hi	Fl. Pt. result in Acc#1		
39	D934	53812	CB5E	52062	DA28	55848	BA28	47656	Multiply from memory location	Addr Lo		Addr Hi	Fl. Pt. result in Acc#1		
40	D9EE	53998	CC18	52248	DAE2	56034	BAE2	47842	Multiply Acc#1 by ten				(result in Acc#1)		
41	DAAE	55982	CCD8	52440	DBA2	56226	BBA2	48034	Unpack memory variable to Acc#1	Addr Lo		Addr Hi			
42	DAE3	56035	CD0D	52493	DBD7	56279	BBD7	48087	Copy Acc#1 to (X,Y) Location	Addr Lo	Addr Hi				
43	DB08	56072	CD32	52530	DBFC	56316	BBFC	48124	Move Acc#2 to Acc#1	(direct call)					
44	DB18	56088	CD42	52546	DC0C	56332	BC0C	48140	Move Rounded Acc#1 to Acc#2	(direct call)					
45	DB1D	56093	CD45	52549	DC0F	56335	BC0F	48143	Move Un-Rounded Acc#1 to Acc#2	(direct call)					
46	DB27	56103	CD51	52561	DC1B	56347	BC1B	48155	Round Acc.#1	(direct call)					
47	DCD9	56537	CF83	53123	DDCD	56781	BDCD	48589	Print fixed-point value	Value Hi	Value Lo				
48	DCE3	56547	CF8D	53133	DDD7	56791	BDD7	48599	Print floating-point value in Acc#1	(direct call)					
49	DCE9	56553	CF93	53027	DDDD	56797	BDDD	48605	Cnvert num to strng at $0100 (calld by 47)	#$00		#$01			
50	FD11	64785	D472	54386	n/a	n/a	n/a	n/a	Entry to M.L.M.	(direct call)					
51	E3D8	58328	E202	57858	E742	59202	E716	59158	Print a character	Char					
52	F156	61782	F185	61829	F1E6	61926	F12F	61743	Print system message			Offset			
53	F0B6	61622	F0D2	61650	EE14	60948	ED09	60681	Send 'talk' to IEEE/Serial	Dev #					
54	F0BA	61626	F0D5	61653	EE17	60951	ED0C	60684	Send 'listen' to IEEE/Serial	Dev #					
55	F128	61736	F143	61763	FF93	65427	FF93	65427	Send secondary address	SA OR $60					
56	F16F	61807	F19E	61742	EEE4	61156	ED40	60736	Send char to IEEE/Serial	Char					
57	F17F	61823	F1AE	61870	EEF6	61174	EDEF	60911	Send 'untalk'	(direct call)					
58	F183	61827	F1B9	61881	EF04	61188	EDFE	60926	Send 'unlisten'	(direct call)					

#	Entry Point For: 2.0		4.0		VIC 20		C64		Operation	Registers In .A	.X	.Y	Registers Out .A	.X	.Y
59	F18C	61836	F1C0	61888	EF19	61209	EE13	60947	Input from IEEE/Serial				Data		
60	F2A9	62121	F2DD	62173	F34A	62282	F291	61985	Close logical file (kernal rtn)	LF #					
61	F301	62209	F335	62261	F770	63344	F6ED	63213	Check for STOP key				Z flag = 1 if pressed		
62	F322	62242	F356	62294	F542	62786	F49E	62510	LOAD subroutine	#$00	Start Lo	Start Hi			
63	F40A	62474	F449	62537	F647	63047	F5AF	62895	Print SEARCHING...		(direct call)				
64	F41D	62493	F45C	62556	F659	63065	F5C1	62913	Print file name		(direct call)				
65	F494	62500	F4D3	62675	F867	63591	F7EA	63466	Find specific tape header block	Len	Pointer to string in $BB, BC (same for 2/4.0)				
66	F5A6	62886	F5E5	62949	F7AF	63407	F72D	63277	Find any tape header block		(direct call)				
67	F812	63506	F857	63575	F894	63524	F817	63511	Press PLAY...; wait		(direct call)				
68	F855	63573	F89A	63530	F8C0	63680	F841	63553	Read tape to buffer		(direct call)				
69	F85E	63582	F8A3	63651	F8C6	63686	F847	63559	Read tape		(direct call)				
70	F886	63622	F8CB	63691	F8E3	63715	F864	63588	Write tape from buffer						
71	F88E	63630	F8D3	63699	F8E8	63720	F869	63593	Write tape, leader length in A	Ldr Len.					
72	FB76	64374	FBBB	64443	FCF6	64758	FB8E	64398	Reset tape I/O		(direct call)				
73	FC9B	64555	FCE0	64736	FCF9	64761	FCBD	64701	Set interrupt vector		(direct call)				

BASIC Keyword Tokens and Entry Points

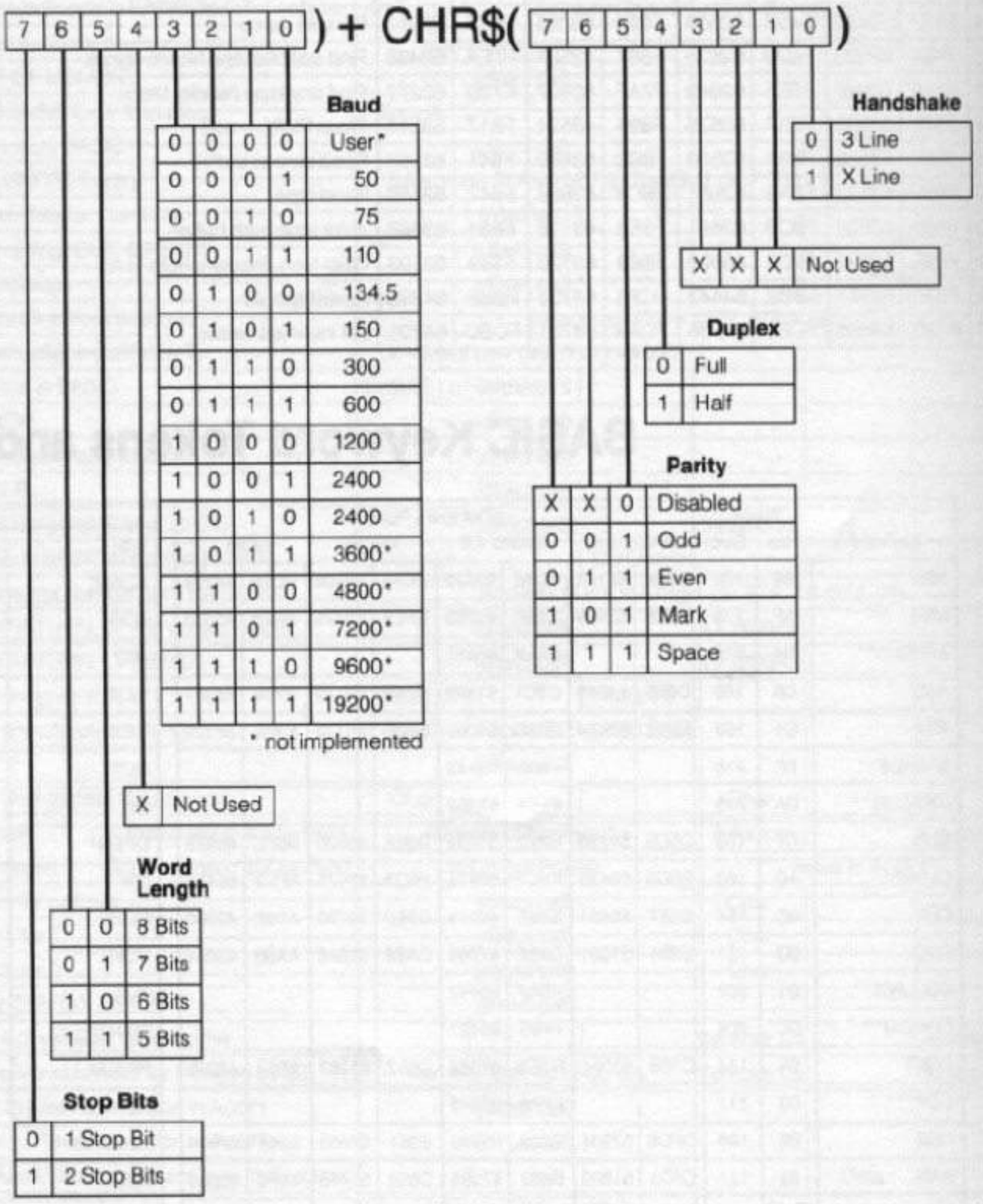
Keyword	Token Hex	Token Dec	ROM Entry Point BASIC 2.0		BASIC 4.0		VIC 20		C64	
ABS	B6	182	DB64	56164	CD8E	52622	DC58	59408	BC58	48216
AND	AF	175	CECB	52939	C089	49289	CFE9	53225	AFE9	45033
APPEND**	D4	212			FFAB	65451				
ASC	C6	198	D665	54885	C8C1	51393	D78B	55179	B78B	46987
ATN	C1	193	E08C	57484	D32C	54060	E30B	58123	E30E	58126
BACKUP**	D2	210			FFA5	65445				
CATALOG**	DA	215			FFB4	65460				
CHR	C7	199	D5C6	54726	C822	51234	D6E6	55020	B6EC	46828
CLOSE*	A0	160	FFC3	65475	FFC3	65475	FFC3	65475	FFC3	65475
CLR	9C	156	C577	50551	B5EE	46574	C65E	50782	A65E	42590
CMD	9D	157	C991	51601	BA8E	47758	CA86	51846	AA86	43654
COLLECT**	D1	209			FFA2	65442				
CONCAT**	CC	204			FF93	65427				
CONT	9A	154	C76B	51051	B7EE	47086	C857	51287	A857	43095
COPY**	D3	211			FFA8	65448				
COS	BE	190	DFD8	57304	D282	53890	E261	57953	E264	57956
DATA	83	131	C800	51200	B883	47235	C858	51448	A8F8	43256
DCLOSE**	CE	206			FF99	65433				
DEF	96	150	D28D	53901	C4DC	50396	D3B3	54195	B3B3	46003
DIM	86	134	CF63	53091	C121	49441	D081	53377	B081	45185
DIRECTORY**	DA	218			FFB4	65460				
DLOAD**	CD	205			FF96	65430				
DSAVE**	D5	213			FFAE	65454				
END	80	128	C741	51009	B7C8	47048	C831	51249	A831	43057
EXP	BD	189	DEDA	57050	D184	53636	DFED	57325	BFED	49133
FN	A5	165	D2CE	53966	C51D	50461	D3F4	54260	B3F4	46068
FOR	81	129	C658	50776	B6DE	46814	C742	51010	A742	42818
FRE	B8	184	D259	53849	C4A8	50344	D37D	54141	B37D	45949
GET*	A1	161	FFE4	65508	FFE4	65508	FFE4	65508	FFE4	65508
GOSUB	8D	141	C790	51088	B813	47123	C883	51331	A883	43139
GOTO	89	137	C7AD	51117	B830	47152	C8A0	51360	A8A0	43168
HEADER**	D0	208			FF9F	65439				
IF	8B	139	C830	51248	B8B3	47283	C928	51496	A928	43304
INPUT*	85	133	FFCF	65487	FFCF	65487	FFCF	65487	FFCF	65487
INPUT#	84	132	CAA7	51879	BBA4	48036	CBA5	52133	ABA5	43941
INT	B5	181	DBD8	56280	CE02	52738	DCCC	56524	BCCC	48332
LEFT	C8	200	D5DA	54746	C836	51254	D700	55040	B700	46848
LEN	C3	195	D656	54870	C8B2	51378	D77C	55164	B77C	46972
LET	88	136	C8AD	51373	B930	47408	C9A5	51621	A9A5	43429

	Token		BASIC 2.0		BASIC 4.0		VIC 20		C64	
LIST	9B	155	C5B5	50613	B630	46640	C69C	50844	A69C	42652
LOAD*	93	147	FFD5	65493	FFD5	65493	FFD5	65493	FFD5	65493
LOG	BC	188	D8F6	55542	C820	52000	D9EA	55786	B9EA	47594
MID	CA	202	D611	54801	C86D	51309	D737	55095	B737	46903
NEW	A2	162	C55B	50523	B5D2	46546	C642	50754	A642	42562
NEXT	82	130	CC20	52256	BD19	48409	CD1E	52510	AD1E	44318
NOT	A8	168	CDCF	52687	BECC	48844	CED4	52948	AED4	44756
ON	91	145	C853	51283	B8D6	47318	C94B	51531	A94B	43339
OPEN*	9F	159	FFC0	65472	FFC0	65472	FFC0	65472	FFC0	65472
OR	B0	176	CEC8	52936	C086	49286	CFE6	53222	AFE6	45030
PEEK	C2	194	D6E8	55016	C943	51523	D60D	55309	B80D	47117
POKE	97	151	D707	55047	C95A	51546	D824	55332	B824	47140
POS	B9	185	D27A	53882	C4C9	50377	D39E	54174	B39E	45982
PRINT*	99	153	FFD2	65490	FFD2	65490	FFD2	65490	FFD2	65490
PRINT#	98	152	C98B	51595	BA88	47752	CA80	51840	AA80	43648
READ	87	135	C807	51975	BC02	48130	CC06	52230	AC06	44038
RECORD**	CF	207			FF9C	65436				
REM	8F	143	C843	51267	B8C6	47302	C93B	51515	A93B	43323
RENAME**	D8	216			FFB7	65463				
RESTORE	8C	140	C730	50992	B7B7	47031	C81D	51229	A81D	43037
RETURN	83	142	C7DA	51162	B85D	47197	C8D2	51410	A8D2	43218
RIGHT	C9	201	D606	54790	C862	51298	D72C	55084	B72C	46892
RND	BB	187	DF7F	57215	D229	53801	E094	57492	E097	57495
RUN	8A	138	C785	51077	B808	47112	C871	51313	A871	43121
SAVE*	94	148	FFD8	65496	FFD8	65496	FFD8	65496	FFD8	65496
SCRATCH**	D9	217			FFBA	65466				
SGN	B4	180	BD45	56133	CD6F	52591	DC39	56377	BC39	48185
SIN	BF	191	DFDF	57311	D289	53897	E268	57960	E26B	57963
SPC(	A6	166	C9FC	51708	BAFD	47869	CAF8	51960	AAF8	43768
SQR	BA	186	DE5E	56926	D108	53512	DF71	57201	BF71	49009
STEP	A9	169	C6AB	50859	B731	46897	C795	51093	A795	42901
STOP	90	144	C73F	51007	B7C6	47046	C82F	51247	A82F	43055
STR	C4	196	D33F	54079	C58E	50574	D465	54373	B465	46181
SYS*	9E	158	F684	63108	F6C3	63171	E127	57639	E12A	57642
TAB(	A3	163	C9FC	51708	BAFD	47869	CAF8	51960	AAF8	43768
TAN	C0	192	E028	57384	D2D2	53970	E2B1	58033	E2B4	58036
USR	B7	183	All ROMs Jump to Location 0000 (thru USR Jump Vector)							
VAL	C5	197	D687	54919	C8E3	51427	D7AD	55213	B7AD	47021
VERIFY*	95	149	FFDB	65499	FFDB	65499	FFDB	65499	FFDB	65499
WAIT	92	146	D710	55056	C963	51555	D82D	55341	B82D	47149

* Kernal Routine / ** BASIC 4.0 Kernal Routine

VIC 20 / Commodore 64 RS 232

OPEN LF, 2, SA, CHR$(Control Register) + CHR$(Command Register (optional))



RS 232 User Port Lines

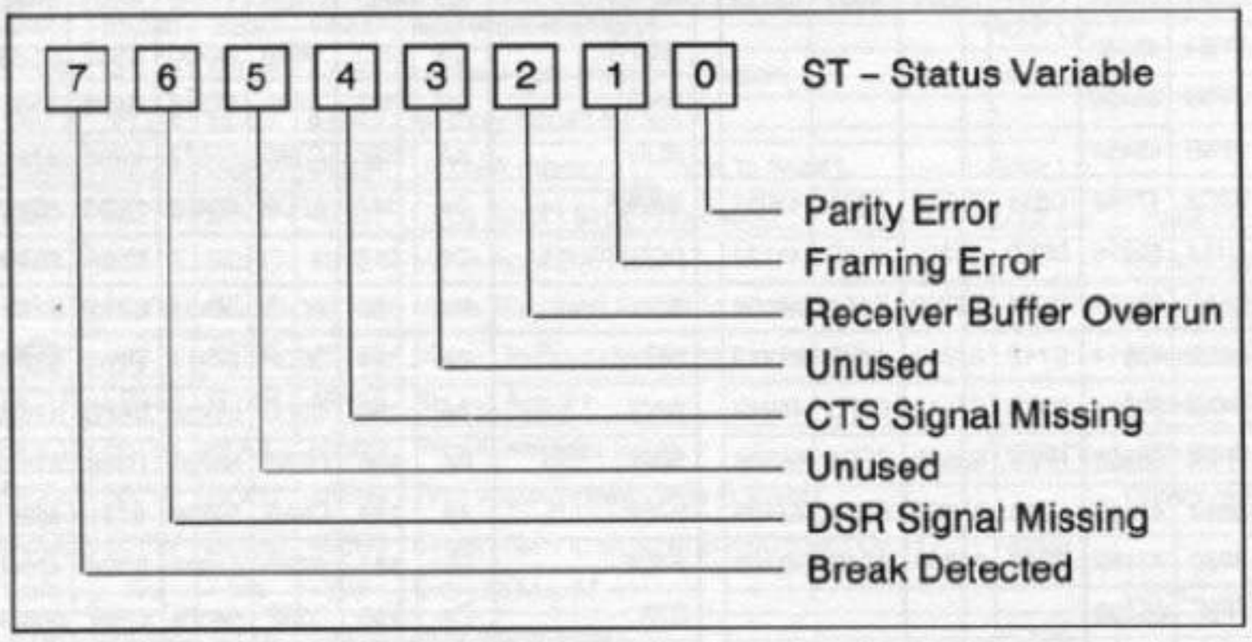
VIC 20 RS 232 is controlled by VIA 1 (6522) at $9110
C64 RS 232 is controlled by CIA 2 (6526) at $DD00

Pin#	Chip	Description	Abrv	Dir.	Modes
A	GND	Protective Ground	GND		1 2
B	FLAG2	Received Data	S_{in}	IN	1 2
C	PB0	Received Data	S_{in}	IN	1 2
D	PB1	Request to Send	RTS	OUT	1* 2
E	PB2	Data Terminal Ready	DTR	OUT	1* 2
F	PB3	Ring Indicator	RI	IN	3
H	PB4	Received line Signal	DCD	IN	2
J	PB5	Unassigned		IN	3
K	PB6	Clear To Send	CTS	IN	2
L	PB7	Data Set Ready	DSR	IN	2
M	PA2	Transmitted Data	S_{out}	OUT	1 2
N	GND	Signal Ground	GND		1 2 3

Available Modes

1) 3-Line interface (S_{in}, S_{out}, GND)
2) X-Line interface
3) User available only (unused in code)
* these lines are held high during 3-line mode.

VIC 20 / C64 RS 232 Status



Notes:

- If the LF# is 128 or greater, a Line Feed will be sent after each Carriage Return
- The Secondary Address SA does not affect RS 232 operation
- Before Closing the channel, check output buffer for data with:

```
VIC 20 : 100 IF ST = 0 AND (PEEK(37151) AND 64) = 64 THEN 100
         110 CLOSE LF
C64 :    100 SX = ST : IF (SX = 0 OR SX = 8) THEN 100
         110 CLOSE LF
```

IEEE 488 Bus Signals

Manager	ATN	Attention	The controller (PET/CBM/B) sets this signal low while it is sending commands on the data bus. When ATN is low, only peripheral addresses and control messages are on the data bus. When ATN is high, only previously assigned devices can transfer data.
Transfer	DAV	Data Valid	When DAV is low, this signifies that data is valid on data bus.
Manager	EOI	End or Identify	When the last byte of data is being transferred, the talker has the option of setting EOI low. The controller always sets EOI low while the last data byte is being transferred from the controller.
Manager	IFC	Interface Clear	The controller sends its internal reset signal as IFC low (true) to initialize all devices to the idle state. When the controller is switched on or reset, IFC goes low for about 100 milliseconds.
Transfer	NDAC	Data Not Accepted	This signal is held low (true) by the listener while reading. When the data byte has been read, the listener sets NDAC high. This signals the talker that data has been accepted.
Transfer	NRFD	Not Ready for Data	When NRFD is low (true), one or more listeners are not ready for the next byte of data. When all devices are ready, NRFD goes high.
Manager	SRQ	Service Request	Not implemented in BASIC, but available to the user.
Manager	REN	Remote Enable	REN is held low by the bus controller. The PET/CBM has a pin grounded that keeps REN permanently low.
Data	D101-8	Data Input/Output Lines 1–8	These signals represent the bits of information on the data bus. When a D10 signal is low, it represents 1 and when high 0.
General	GND	Ground	Ground connections: There are six control and management signal ground returns, one data signal ground return and one chassis shield ground lead.

Cassette Port

Pin#	Name	Description
A–1	GND	Digital Ground
B–2	+5V	+5 Volts to operate cassette circuitry only
C–3	Motor	Computer controlled +6V for cassette motor
D–4	Read	Read line from cassette
E–5	Write	Write line cassette
F–6	Sense	Monitors closure of any locking type cassette switch

Note: Upper and Lower cassette pins are shorted

IEEE Port Pinouts

Pin #	Pin#*	Mnemonic	Definition
1	1	DIO1	Data Input/Output Line #1
2	2	DIO2	Data Input/Output Line #2
3	3	DIO3	Data Input/Output Line #3
4	4	DIO4	Data Input/Output Line #4
5	5	EOI	End or Identify
6	6	DAV	Data Valid
7	7	NRFD	Not Ready For Data
8	8	NDAC	Data not Accepted
9	9	IFC	Interface Clear
10	10	SRQ	Service Request
11	11	ATN	Attention
12	12	GND	Chassis Ground (IEEE cable shield)
A	13	DIO5	Data Input/Output Line #5
B	14	DIO6	Data Input/Output Line #6
C	15	DIO7	Data Input/Output Line #7
D	16	DIO8	Data Input/Output Line #8
E	17	REN	Remote Enable
F	18	GND	DAV Ground
H	19	GND	NRFD Ground
J	20	GND	NDAC Ground
K	21	GND	IFC Ground
L	22	GND	SRQ Ground
M	23	GND	ATN Ground
N	24	GND	Data Ground (DIO1–8)

* Pin Numbers for Standard IEEE Cable Connector

PET/CBM User Port

Pin#	Function	Description
1	Ground	System Ground
2	TV Video	Video Out for external displays
3	SRQ	Connected to IEEE SRQ
4	EOI	Connected to IEEE EOI
5	Diag Sense	Held low causes power up to Diagnostic routines
6	READ 1	Connected to cassette 1 read line
7	READ 2	Connected to cassette 2 read line
8	Write	Diagnostic tape write verify
9	Vert	TV Vertical for external displays
10	Horiz	TV Horizontal for external displays
11	GND	
12	GND	
A	GND	
B	CA1	Edge sensitive input of 6522 VIA
C	PB0	PB0–7 are independently programmable
D	PB1	for Input or Output
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	CB2	Special I/O pin of VIA
N	GND	Digital Ground

VIC 20 Expansion Port

Pin#	Name	Description
1	GND	System ground
2	CD0	Data bus bit 0
3	CD1	Data bus bit 1
4	CD2	Data bus bit 2
5	CD3	Data bus bit 3
6	CD4	Data bus bit 4
7	CD5	Data bus bit 5
8	CD6	Data bus bit 6
9	CD7	Data bus bit 7
10	BLK1	8k decoded RAM/ROM block 1 @ $2000 (active low)
11	BLK2	8k decoded RAM/ROM block 2 @ $4000 (active low)
12	BLK3	8k decoded RAM/ROM block 3 @ $6000 (active low)
13	BLK5	8k decoded ROM block 5 @ $A000 (active low)
14	RAM1	1k decoded RAM block @ $0400 (active low)
15	RAM2	1k decoded RAM block @ $0800 (active low)
16	RAM3	1k decoded RAM block @ $0C00 (active low)
17	V R/W	Read/Write line from VIC Chip (high-read, low-write)
18	C R/W	Read/Write line from CPU (high-read, low-write)
19	IRQ	Interupt Request line to 6502 (active low)
20	NC	
21	+5v	
22	GND	
A	GND	
b	CA0	Address bus bit 0
C	CA1	Address bus bit 1
D	CA2	Address bus bit 2
E	CA3	Address bus bit 3
F	CA4	Address bus bit 4
H	CA5	Address bus bit 5
J	CA6	Address bus bit 6
K	CA7	Address bus bit 7
L	CA8	Address bus bit 8
M	CA9	Address bus bit 9
N	CA10	Address bus bit 10
P	CA11	Address bus bit 11
R	CA12	Address bus bit 12
S	CA13	Address bus bit 13
T	I/O 2	I/O block 2 (located at $9600)
U	I/O 3	I/O block 3 (located at $9C00)
V	Φ02	Phase 2 system clock
W	NMI	6502 Non-Maskable Interrupt (active low)
X	RESET	6502 Reset pin (active low)
Y	NC	
Z	GND	

VIC 20 User Port

Pin#	Function	Description
1	Ground	System Ground
2	+5V	(100 ma maximum)
3	RESET	Cold Start. Memory is destroyed
4	JOY 0	Joystick Switch 0
5	JOY 1	Joystick Switch 1
6	JOY 2	Joystick Switch 2
7	PEN	Light Pen Input. Also Joystick Fire Button
8	SENSE	Cassette Switch sense line
9	Serial ATN	Connected to Serial Bus ATN Line
10	9 VAC + Phase	Transformer output (50 ma. maximum)
11	GND	
12	GND	
A	GND	
B	CB1	
C	PB0	PB0–7 are independently programmable
D	PB1	for Input or Output
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	CB2	Special I/O pin of VIA
N	GND	

VIC 20 Audio/Video Port

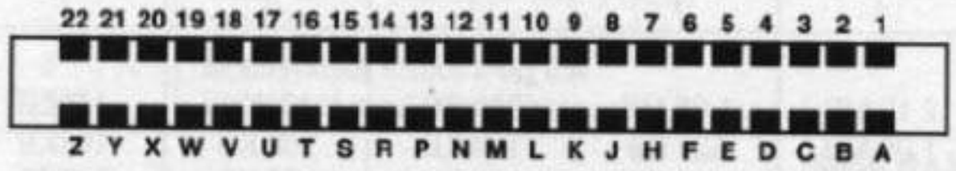
Pin#	Name	Description	Colour
1	+5V	10 ma. maximum	Red
2	GND	System Ground	-
3	AUD	Audio Out	Grey
4	VID L	Video Low	Black
5	VID H	Video High	White

Colour refers to Radio Shack Part# 42-2394

VIC 20 / Commodore 64 Serial Port

Pin#	Name	Description
1	SRQ	Serial SRQ in (active low)
2	GND	System Ground
3	ATN	Serial ATN In/Out
4	CLK	Serial Clock In/Out
5	DATA	Serial Data In/Out
6	RESET	Resets all devices on Serial bus (active low)

Commodore 64 Expansion Port



Name	Pin#	Description
1	GND	System Ground.
2	+5 VDC	Total User Port and Cartridge devices
3	+5 VDC	can draw no more than 450ma.
4	IRQ	Interupt Request line to 6502 (active low).
5	R/W	Read/Write.
6	Dot Clock	8.18 MHz video dot clock.
7	I/O 1	I/O Block 1 @ $DE00-$DEFF (active low) unbuffered I/O.
8	GAME	Active low TTL input.
9	EXROM	Active low TTL input.
10	I/O 2	I/O Block 2 @ $DF00-$DFFF (active low) buffered TTL output.
11	ROM L	8K decoded RAM/ROM block @ $8000 (active low) buffered TTL output.
12	BA	Bus Available signal from the VIC II chip - unbuffered - 1 is maximum load.
13	DMA	Direct Memory Access request line (active low input) is TTL input.
14	D7	Data bus bit 7 *
15	D6	Data bus bit 6 *
16	D5	Data bus bit 5 *
17	D4	Data bus bit 4 *
18	D3	Data bus bit 3 *
19	D2	Data bus bit 2 *
20	D1	Data bus bit 1 *
21	D0	Data bus bit 0 *
21	GND	System ground.
A	GND	System Ground
B	ROM H	8K decoded RAM/ROM Block @ $E000 buffered.
C	RESET	6502 RESET pin (active low) buffered TTL out/unbuffered in.
D	NMI	6502 Non-Maskable Interrupt (active low) buffered TTL out, unbuffered in.
E	Φ2	Phase 2 system clock.
F	A15	Address bus bit 15 *
H	A14	Address bus bit 14 *
J	A13	Address bus bit 13 *
K	A12	Address bus bit 12 *
L	A11	Address bus bit 11 *
M	A10	Address bus bit 10 *
N	A9	Address bus bit 9 *
P	A8	Address bus bit 8 *
R	A7	Address bus bit 7 *
S	A6	Address bus bit 6 *
T	A5	Address bus bit 5 *
U	A4	Address bus bit 4 *
V	A3	Address bus bit 3 *
W	A2	Address bus bit 2 *
X	A1	Address bus bit 1 *
Y	A0	Address bus bit 0 *
Z	GND	System Ground

* = Unbuffered, 1 is TTL load max.

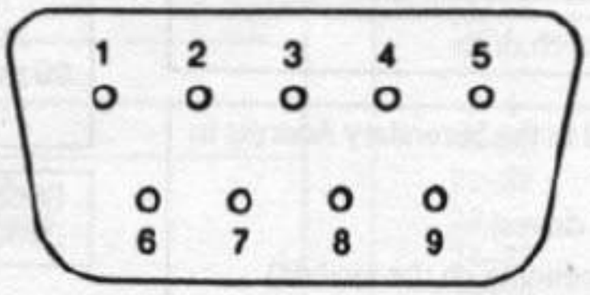
Commodore 64 User Port

Pin#	Function	Description
1	Ground	System Ground
2	+5V	(100 ma maximum)
3	RESET	Cold Start. Memory is NOT destroyed
4	CNT1	Serial Port counter from CIA #1
5	SP1	Serial Port from CIA #1
6	CNT2	Serial Port counter from CIA #2
7	SP2	Serial Port from CIA #2
8	PC2	Handshaking line from CIA #2
9	Serial ATN	Connected to Serial Bus ATN Line
10	9 VAC +Phase	Transformer output (50 ma. maximum)
11	9 VAC –Phase	Transformer output (50 ma. maximum)
12	GND	
A	GND	
B	FLAG2	
C	PB0	PB0–7 are independently programmable
D	PB1	for Input or Output
E	PB2	
F	PB3	
H	PB4	
J	PB5	
K	PB6	
L	PB7	
M	PA2	Special I/O pin of CIA
N	GND	

Commodore 64 Audio/Video Port

Pin#	Name	Description
1	LUM	Luminance
2	GND	System Ground
3	AUD	Audio Out
4	COMP	Composite Video
5	JACK	Audio In
6	CHR	Chroma out
7	N/C	No connection
8	N/C	No connection

VIC 20 / Commodore 64 Joystick Ports



Pin#	Name	Description
1	JOY 0	
2	JOY 1	
3	JOY 2	
4	JOY 3	
5	POT Y	
6	FIRE	Also the Light Pen input. (C64 port 1 only)
7	+5V	100 ma. maximum
8	GND	System Ground
9	POT X	

Note: See Memory Map for reading Joystick Ports

Commodore Disk Specifications

Model	D9090	D9060	8250	8050	4040	2031	1541
Drives per Head	1	1	2	2	2	1	1
Heads per Drive	6	4	2	1	1	1	1
Formatted Storage							
Capacity per Unit	7.47 MB	4.98 MB	2.12 MB	1.05 MB	340 KB	170KB	170KB
Max Sequential Files/Drive	7.41 MB	4.94 MB	1.05 MB	521 KB	168 KB	168 KB	168KB
Max Relative Files/Drive	7.35 MB	4.90 MB	1.04 MB	183 KB	167 KB	167 KB	167KB
Disk System Buffer	4 KB	4 KB	4 KB	4 KB	4 KB	2 KB	2KB
Disk Formats							
Cylinders (Tracks)	153	153	77	77	35	35	35
Sectors per Cylinder	128	192	–	–	–	–	–
Sectors per Track	32	32	23–29	23–29	17–21	17–21	17–21
Bytes per Sector	256	256	256	256	256	256	256
Blocks Free	29162	19442	8266	4104	1328	664	664
Transfer Rates (bytes per second)							
Internal to Unit	5 MB	5 MB	40 KB	40 KB	40 KB	40 KB	–
IEEE–488 Bus	1.2 KB	1.2 KB	1.2 KB	1.2 KB	1.2 KB	1.2 KB	–
Access Times (milli-seconds)							
Track-To-Track	3	3	5	*	30	30	30
Average Track	153	153	125	**	360	360	360
Head Settling Time	15	15	–	–	–	–	–
Average Latency	8.34	8.34	100	100	100	100	100
RPM	3600	3600	300	300	300	300	300
* Track-To-Track: Micropolis 8050 = 30 ms. Tandon 8050 = 5 ms. ** Average Track: Micropolis 8050 = 750 ms. Tandon 8050 = 125 ms.							
Physical Dimensions							
Height (inches)	5.75	5.75	7.0	7.0	7.0	5.5	3.0
Width (inches)	8.25	8.25	15.0	15.0	15.0	8.0	7.0
Depth (inches)	15.25	15.25	13.75	13.75	13.75	14.25	13.0
Weight (pounds)	21	21	28	28	28	20	10
Electrical							
Power (Watts)	200	200	60	50	50	40	35
Voltage (all models)	110 – 120 VAC. 60 Hz						

Disk Utility-Command Set

Command	Abbreviation	Format
Block-Read	B-R	"B-R:" ch;dr;t;s
Block-Write	B-W	"B-W:" ch;dr;t;s
Block-Execute	B-E	"B-E:" ch;dr;t;s
Buffer-Pointer	B-P	"B-P:" ch;p
Block-Allocate	B-A	"B-A:" dr;t;s
Block-Free	B-F	"B-F:" dr;t;s
Memory-Write	M-W	"M-W" adL/adH/nc/data
Memory-Read	M-R	"M-R" adl/adh
Memory-Execute	M-E	"M-E" adl/adh
User Command	U	"ux:ch;dr;t;s

CH	The channel number in DOS: identical to the Secondary Address in the associated OPEN statement
DR	The Drive number: 0 (or 1 floppy dual drives)
T	The Track number: 1 through 154 (depending on the model*)
S	Sector number : 0 through 112 (depending on the model*)
P	The pointer Position for the buffer pointer
ADL	The Low byte of the Address (use CHR$(ADL))
ADH	The High byte of the Address (use CHR$(ADL))
NC	The Number of Characters: 1 through 34
DATA	The actual data in hexadecimal. This is transmitted by using the CHR$ function, ie. CHR$(17) would send the decimal equivalent of hexadecimal 11
X	The index to the user table
PARMS	The Parameters associated with the U command (optional)

Sector Distribution By Track

	Number of Sectors		
Track Number	4040	2031	1541
1 – 17	21	21	21
18 – 24	19	19	19
25 – 30	18	18	18
31 – 35	17	17	17

Track Number	8050	8250	
1 – 39	29	29	
40 – 53	27	27	
54 – 64	25	25	
65 – 77	23	23	
78 – 116		29	
117 – 130		27	
131 – 141		25	
142 – 154		23	

D9060/D9090 – 153 tracks per recording surface (4 on D9060 and 6 on the D9090) with 32 sectors per track

User Command Jump Table

Standard Syntax	Alternate Syntax	Function
U1	UA	Block-Read replacement
U2	UB	Block-Write replacement
U3	UC	Jump to $1300
U4	UD	Jump to $1303
U5	UE	Jump to $1306
U6	UF	Jump to $1309
U7	UG	Jump to $130C
U8	UH	Jump to $130F
U9	UI	Jump to $10F0 (NMI)
U:	UJ	Power-Up vector (reset)

BAM (Block Allocation Map) Formats

4040, 2031, and 1541 BAM Format – Track 18 Sector 00					
Byte#	Description	Data			
0–1	Track–Sector of first Directory block	18–00			
2	ASCII 'a' identifies DOS 2.6 format	65			
3	Reserved for future DOS use	00			
4–143	Bit map of available blocks	tracks 1–35			

8050 BAM Format					
Byte#	Description	Data			
		BAM 1 Tr38 / Sc00	BAM 2 Tr38 / Sc03		
0–1	Track–Sector of next BAM block	38–03	39–01		
2	ASCII 'c' identifies DOS 2.5 format	67	67		
3	Reserved for future DOS use	00	00		
4	Lowest track # mapped in this BAM block	01	51		
5	Highest track # (+1) mapped in this BAM block	51	78		
6	Number of unused blocks on track:	1	51		
7–10	Bit map of available blocks on track:	1	51		
11–255	(BAM 2: 11–140)Bit map of available blocks on tracks:	2–50	52–77		

8250 BAM Format					
Byte#	Description	Data			
		BAM 1 Tr38 / Sc00	BAM 2 Tr38 / Sc03	BAM 3 Tr38 / Sc06	BAM 4 Tr38 / Sc09
0–1	Track–Sector of next BAM block	38–03	38–06	38–09	38–01 (Dir)
2	ASCII 'c' identifies DOS 2.7 format	67	67	67	67
3	Reserved for future DOS use	00	00	00	00
4	Lowest track # mapped in first BAM block	01	51	101	151
5	Highest track # (+1) mapped in first BAM block	51	101	151	155
6	number of unused blocks on track:	1	51	101	151
7–10	bit map of available blocks on track:	1	51	101	151
11–255	(BAM 4: 11–25)Bit map of available blocks on tracks:	2–50	52–100	102–150	152–154

D9060 / D9090 BAM Format – Track 1 Sector 0 (normal location)					
Byte#	Description	Data			
0–1	Track–Sector pointer to next BAM block	$FFFF=last			
2–3	Track–Sector pointer to previous BAM block	$FFFF=first			
4	Lowest track # mapped in this BAM block				
5	Highest track # (+1) mapped in this BAM block				
6	Number of blocks unused on this track				
7–10	Bit map of available blocks on this track				
11–255	Bit map of the next 49 tracks				

Directory Format

2031, 4040, 1541 Directory Header – Track 18 Sector 00		
Byte#	Data	Description
1–143		Reserved for 2031 BAM
144–161		Diskette name, padded with shifted spaces
162–163		Diskette ID number
164	160	Shifted space
165–166	50, 65	ASCII '2a' identifies DOS version and format
167–170	160	Shifted spaces
171–255	00	Not used

8050, 8250 Directory Header – Track 39 Sector 00		
Byte#	Data	Description
0–1	38, 00	Track–Sector to first BAM block
2	67	ASCII 'c' identifies DOS 2.5 format
3	00	reserved for future DOS use
4–5		Not used
6–21		Diskette name, padded with shifted spaces
22–23	160	Shifted spaces
24–25		Diskette ID number
26	160	Shifted space
27–28	50, 67	ASCII '2c' identifies DOS version and format
29–32	160	Shifted spaces
33–255	00	Not used

D9060 / D9090 Directory Header – Track 0 Sector 0		
Byte#	Data	Description
0–1		Track–Sector pointer to bad track and sector list
2–3	00,255	Identifies DOS 3.0 format
4–5	76, 00	Track–Sector of first directory block
6–7	00, 00	Not used
8–9	01, 00	Track–Sector of first BAM block

2031 Directory Blocks - Track 18 Sector 01 through 18

4040 Directory Blocks - Track 18 Sector 01 through 18

8050 Directory Blocks - Track 39 Sector 01 through 29

8250 Directory Blocks - Track 39 Sector 01 through 29

D9060 / D9090 Directory Blocks - Starting on cylinder 76, uses all Tracks - Sectors 00 through 31, then expands to additional blocks as required, providing 'unlimited' Directory size.

Byte#	Description
0–1	Track–sector pointer to next directory block
2	File type
3–4	Track–sector pointer to first file block
5–20	File name, padded with shifted spaces
21–22	Track–sector of first side sector if RELative file
23	Record length if relative file
24–27	Reserved for future file information
28–29	Track–sector pointer for replacement
30–31	Number of blocks used by the file
32–255	Seven more 32–byte file entries (same as 2–31 above, plus two additional unused bytes)

Additional Notes	
1	32 bytes per file entry, except the first entry is 30 bytes
2	Total of eight (8) file entries per directory block
3	File types are: Scratched Files $00 Sequential Files $01 Program Files $02 User–Defined $03 Relative Record $04
4	File type codes are OR'ed with $80 when file is properly closed
5	Track value of 00 in byte zero indicates the last used block in the directory. Sector value then shows next byte to use

Commodore Related Programming Books

Title	Author	Publisher
SuperPET Related Books.		
SuperPET System Overview	Boswell/Grove/McPhee/Welch	Sam's Books
SuperPET Waterloo 6809 Assembler	Cowan/Shaw	Sam's Books
SuperPET Waterloo MicroBASIC	Graham/McPhee	Sam's Books
SuperPET Waterloo MicroFORTRAN	Dirkson/Welch	Sam's Books
SuperPET Waterloo MicroCOBOL	Commodore	Sam's Books
SuperPET Waterloo MicroPASCAL	Boswell/Grove/Welch	Sam's Books
SuperPET Waterloo MicroAPL	Wilson/Wilkerson	Sam's Books
6502 Programming Books.		
6502 Assembly Language Programming	Lance Leventhal	Osborne
6502 Assembly Language Subroutines	Lance Leventhal	Osborne
6502 Software Gourmet Guide & Cookbook	Findley	?
Advanced 6502 Programming	Zaks	Sybex
Programming The 6502	Zaks	Sybex
6502 Software Design	Scanlon	Sams
6502 Assembly Language Programming	Fernandez/Tabler/Ashley	WILY
6502 Systems Programming	Windeknecht	LB
Beyond Games: Systems Software for your 6502 Personal Computer	Skiers	BYTE
Top Down Assembly Language Programmer For Your 6502 Personal Computer	Skier	McGraw Hill
Machine Language For Beginners	Mansfield	Compute!
6809 Programming Books.		
6809 Assembly Language Programming	Lance Leventhal	Osborne
Programming The 6809	Zaks/Labiak	?
The 6809 Companion	James	BP
Hardware Related Books.		
PET/CBM & The IEEE 488 BUS (GPIB)	Fisher/Jensen	Osborne
Interface Projects For The PET/CBM	Hallgren	?
PET Interfacing	Downey/Rogers	Sams
Advanced 6502 Interfacing	Holland	Sams
Programming & Interfacing the 6502 with experiments	De Jong	Sams
How To Program and Interface the 6800	Staugaard	Sams
6809 Microcomputer Programming and Interfacing, with Experiments	Staugaard	Sams
6502 Applications	Zaks	Sybex

Title	Author	Publisher
Vic 20 Books.		
Vic Programmers Reference Guide	...	Sams
Compute's First Book Of Vic	Compute! Books	Compute! Books
Compute's Second Book Of Vic	Compute! Books	Compute! Books
Compute's First Book Of Vic Games	Compute! Books	Compute! Books
Users Handbook to the Commodore Vic 20	Weber	WSI
Vic-20 Computer Writing Workbook	Adler	ARCsoft
34 Vic-20 Computer Programs for Home, School & Office	Adler	ARCsoft
101 Programming Tips & Tricks For The Vic-20	Adler	ARCsoft
Start With BASIC On Your Commodore Vic 20	Monro	REST
Vic BASIC: A User-Friendly Guide	Zamora	PH
More Than 32 BASIC Programs for the Commodore Vic Computer	Rugg/Feldman/Wilson	Dilithium Press
Computers for Kids-Commodore Vic-20 Edition	Larsen	Creative Comp. Press
Getting Aquainted with your Vic-20	Hartnell	Creative Comp. Press
Vic Graphics	Nick Hampshire	Hayden
Vic Revealed	Nick Hampshire	Hayden
Vic Games	Nick Hampshire	Hayden
Mapping The Vic	Compute!	Compute! Books
C64 Books.		
Commodore 64 Programmers Ref.Guide	Commodore	Sams
Compute!s First Book of 64	Compute!	Compute! Books
More Than 32 BASIC Programs for the Commodore 64 Computer	Rugg/Feldman/Moore	Dilithium Press
Mapping The Commodore 64	Compute!	Compute! Books
Pet/CBM Books.		
Programming The PET/CBM	Raeto Collins West	Compute Books
The PET Revealed	Nick Hampshire	Hayden
The Library Of PET Subroutines	Nick Hampshire	Hayden
PET Graphics	Nick Hampshire	Hayden
CBM Professional Computer Guide	Osborne	Osborne
PET Personnel Computer Guide	Osborne	Osborne
Hands-On BASIC With PET	Peckham	Osborne
Compute's First Book Of PET/CBM	Compute! Books	Compute! Books
PET Basics	David	Dilithium Press
PET Fun and Games	Jeffries and Fisher	?
Mostly BASIC: Applications for your PET	Berenbon	Sams
PET BASIC 1	Zamora	PH
32 BASIC Programs for the PET Computer	Rugg	Dilithium Press
PET/CBM An Introduction To BASIC Programming and Applications	Streitmatter/Goldstein	BRDY
PET/CBM BASIC	Haskell	PH
PET Fun And Games	Jeffries	Osborne
A PET In The Classroom	Boren	Dilithium Press
Instant BASIC	Brown	?

Glossary

Access Time	– The time required for a computer to locate and transfer data to or from a storage medium.
Acoustic Coupler	– The two rubber cups on an acoustic modem that hold the receiver in place to allow transmission and reception of data.
Acoustic Modem	– A telephone modem that is not hardwired into the telephone lines, but uses a microphone and speaker to transmit data to and from the telephone.
Accumulator	– A location within the microprocessor that is used for storage of the result of an arithmetic or logic operation.
Address	– A location between 0 and 65535 in computer memory.
Algorithm	– A set of rules, usually mathematical, that help in solving a particular problem.
ANSI	– American National Standards Institute.
APL	– Advanced Programming Language.
ASCII	– The 'American Standard Code for Information Interchange'. Is an eight level code serial transmission of alphanumeric code. The first 7 bits represent 128 standard ASCII characters. The eighth bit is a parity bit for error checking. The eight data bits are preceded by one start bit and and followed by one or two stop bits.
Assembler	– A computer program that translates source code, ie. the actual written assembler program, into a machine readable form, the object code.
Backup	– The exact duplication of a diskette by the use of a dual disk drive and the backup procedure.
Bank Switching	– A method in which switching between various RAM locations can be achieved through software.
BASIC	– Acronym for Beginners All–purpose Symbolic Instruction Code.
Baud Rate	– The rate of which data is transmitted, which is expressed in terms of bits per second. Baud rate/10 equals bytes per second (approx.)
Binary Number	– A numbering system that uses either the digits 0 or 1 to express all numeric values. The binary digit is also known as the 'bit'.
Bit	– One eighth of a normal byte, in binary it is either a 0 or a 1 symbolizing yes or no, on or off, etc.
Block	– 256 characters of information, as stored on a diskette. Also known as a sector of a diskette.
Buffer	– RAM used for temporary storage of data until the data can be accepted by the device it is being sent to.
Bus	– The lines connecting the memory, microprocessors, and other portions of the computer or peripherals that send or receive data.
Byte	– The equal of one character of information, which is 8 bits. Numeric value of one byte can fall between 0 and 255 decimal.
CPU	– Central Processing Unit, the microprocessor.
COBOL	– COmmon Business Oriented Language.
COMAL	– COmmon Algorithmic Language. Written for Commodore machines in Denmark.
Compiler	– A computer program that will accept a program written in a language such as BASIC, then convert it over to a form that the computer will immediately understand, a machine language.
Daisy Wheel	– A print wheel that uses impact of a solid character to produce its impression. The daisy wheel is circular in design to allow for the wheel to spin around quickly to the correct character.
Density	– The number of bits in a single linear track measured per unit of length of the recording medium, ie. single or double density diskettes.
Data Transmission Rate	– The BAUD rate.
Direct Access	– A method of accessing information from a storage medium, such as a diskette, where the access time is not affected by the location of the data on the disk.

Glossary

Directory	– A list generated by the disk drive of all files currently stored on its diskette.
Diskette	– A thin, flexible magnetic disk permanently enclosed in a semirigid protective jacket.
DOS	– The 'Disk Operating System' which controls all of the actions of the disk drive.
Dot Matrix Printer	– A printer head that forms it characters by the use of numerous metal pins to punch out the image required on paper.
Dynamic memory	– A form of random access memory that maintains its stored information by constant refresh cycles.
Dynamic	– In a constant state of change. (ie. variables)
EPROM	– Erasable Programmable Read Only Memory. Is a form of ROM that can be erased by ultraviolet light, and re-programmed when necessary.
EAROM	– Electrically Alterable Read Only Memory. Is a form of ROM that will remain constant under normal operating conditions, but whose contents can be modified by a series of deliberate electrical signals.
File	– A collection of related records brought together as one, single unit.
File Type	– Pertaining to Commodore DOS, there are four different file types. These are PRoGram, SEQuential, RELative, and USeR.
Flag	– A single storage bit used to signal the occurence of a certain condition.
Floppy Disk	– A diskette
Flow Chart	– A logical diagram of the order in which a particular problem is thought to be resolved. Usually required when complex computer programs are to be written.
Formatting	– Pertaining to Commodore DOS, is the process of organizing a diskette into specific tracks and sectors to allow for storage and retrieval of information.
FORTRAN	– An acronym for FORmula TRANslating system, a high level computer language.
Full Duplex	– In communications, the simultaneous transmission and reception of data, unlike half-duplex.
Graphic Plotter	– A device that connects to the computer, and is controlled by the computers software to draw exact patterns as determined as the computed X and Y co-ordinates.
Half-Duplex	– In communications, allowing only one way data transmission or reception at any one single time. Compares in principal to an inexpensive walkie talkie.
Hard-Copy	– Refers to any printed output in general.
Hard Disk	– A totally self enclosed data storage system. The disk itself is sealed inside the unit, and rotates at approximately 3500 RPM to allow for high speed access to information. A special feature of this type of storage system is the vast amount of storage space available in relation to a floppy disk system.
Hardware	– The actual physical components of a computer system, ie. computer, peripherals, magnetic storage media, paper etc.
Hexadecimal	– A numbering system with the base 16. Its digits range from 0 through F.
High-Level Language	– A computer language such as BASIC, COBOL, FORTRAN, PASCAL, APL and PL/1, to name a few, that use english like commands in the design of computer programs.
Impact printer	– A printer which forms its characters by the physical striking of the ribbon and paper.
Increment	– To increase the value of by a certain quantity.
Interactive language	– A high-level language that allows execution, modification, then re-execution again, for as many cycles as required. BASIC is one example of such a language.
Interface	– A wiring device to connect two or more devices together in a harmonious existence, where data can be transmitted or received accurately.
Interpreter	– A computer program that translates and executes each statement in a program entirely before attempting to continue on to the next statement in line. This is a built in feature of Commodore BASIC.

Glossary

Interrupt Driven	– A program whose execution is activated by the computers operating systems attempts to check its various interrupt routines. Similar to a water wheel placed in a fast moving river. Very little change in the flow if the river but the water wheel is allowed to operate.
I/O	– An abbreviation for Input/Output, or the transmission to and from the microprocessor.
Jump	– In assembler, a branch to a section of code other then what the normal program flow would expect.
K	– In reference to storage capacity, it is equal to 1024 bytes of data.
Light Pen	– A hand–held, light–sensitive device that is wired directly to the computer to allow the light–pen/screen positions to be transmitted to the computer for analysis and re–action.
Low–Level Language	– A computer language, such as assembler, that requires very little translation to become acceptable by the computer.
Machine Language	– A programming language that is immediately understandable by the computer, therefore not requiring any translation at all. In reality not a language as such but a coding of binary digits.
Matrix Printer	– Dot Matrix Printer.
Memory Map	– A listing of addresses and corresponding functions for all RAM and ROM locations within the computers memory.
Microprocessor	– The central processing unit of a microcomputer containing the control unit, arithmetic logic unit, and a number of storage location, all within one small silicon chip.
Modem	– A device that MOdulates and DEModulates signals transmitted over the telephone lines, from analog to digital and vice–versa.
Monitor	– A video–display, similar to a television screen, that is used for computers and other electronic devices requiring a stable video output.
Mnemonic Code	– A set of abbreviated commands designed to help make the writing of an assembly language program simpler for the programmer.
Network	– A number of computers and peripherals connected together through either a form of timesharing system, or over the telephone lines in a similar manner.
Nibble	– A group of 4 bits, or one half of a byte.
Null String	– A string variable without any characters in it.
Object Code	– The compiled, or machine code version of what was originally a source code program.
Operand	– The address of a unit of data to be operated upon.
Operating system	– The software that controls the execution of all computer programs, the diagnostic system to help in the debugging process, all input/output, all housekeeping duties of the computer, and all else that go into the running of the system. In Commodore, the operating system is located in ROM.
Output device	– A device, or one in a series of devices used to collect data from another device. To alleviate confusion, the printer is an output device when accepting data from the computer to be printed.
Pascal	– A high–level programming language.
Parallel	– A type of port or interface that allows all eight bits of data to be transmitted at the same time, each through a separate piece of wire.
Peek	– A BASIC keyword used to look at a position in memory to determine the value located there.
Peripheral	– Any number of external devices that can be used with the computer to increase its productivity.
Pixel	– One graphic point on the computers screen. The type of computer, the graphic mode of operation it is in, and the resolution capabilities of the screen determine how high of resolution will be viewed.
Poke	– A BASIC keyword used to store a numeric digit between the values of 0 and 255, within memory.
Port	– A connection point on a computer to connect an external interface to with some form of peripheral.

Glossary

PROM	– Programmable Read Only Memory. A memory storage chip that may be programmed by someone other than the manufacturer. This chip can only be written to once, and becomes a permanent modification when the process is complete.
QWERTY	– On a full sized typewriter keyboard, or computer terminal, the first six letters in the third row are QWERTY
RAM	– Random Access Memory. As the name implies, this memory is such that access to it can be made in a total random sequence. When the power is turned off with this type of memory, all contents are lost.
Read	– The operation performed by the computer as it takes in new information from an input device.
Read/Write Head	– The piece of hardware that can take information from, or write information to a magnetic storage medium.
Register	– A number of small temporary storage locations within the computer, that are used for such things as the operating system, and other functions that the programmer may require.
Resolution	– The number of pixels (points) that can be displayed on a computer screen both horizontally and vertically.
ROM	– Read Only Memory. As the name implies, the contents of this section of memory can be read from, but not written to. When the power is turned off, the contents of ROM are retained.
Sector	– An area on diskette that is comprised of 256 characters of information. Also know as the block, this block is read and write compatible, and can be accessed by the experienced programmer, or left up to the devices of the disk drive itself.
Sequential Access	– A manner in which to read or write data as complete blocks of information, with each byte accessed in the exact order in which was stored.
Serial	– A type of port or interface that allows only one bit of data to be transmitted at a time through one the data line.
Software	– A computer program and its supporting documentation.
Source Code	– A high level language program that requires interpretation or translation into a form acceptable by the computer before processing.
Static Memory	– A form of RAM whose contents are stable as long as the computer is turned on. Unlike dynamic RAM, there is no refresh cycle required.
Storage Capacity	– The amount of data that a device can accept for storage, usually expressed in bytes.
String	– A set of alphanumeric characters stored within memory that can be recalled, modified, or transmitted to peripheral device such as a printer, disk drive, cassette drive, modem, etc.
Syntax	– The correct grammatical design of a computer program according to the particular rules set down when the computer language was designed.
Synchronous	– Occurring simultaneously with a regular or predictable time relationship.
Thermal Printer	– A dot–matrix printer producing its printed characters by the use of a dot–matrix pattern that has been adapted electrically and heat sensitive paper.
Track	– With reference to a diskette, a track is one circular portion of a diskette surface that is further subdivided into sectors, or blocks of information.
Variable	– An area of identifiable storage space that can be read or altered as necessary. Variable length cannot exceed 255 characters.
Volatile Memory	– A form of computer memory that loses its contents after the power has been turned off. Random access memory, or RAM is an example of this.
Write	– A store a section of computer memory, or a quantity of produced data on an external storage medium such as diskette or cassette.
Write Protect	– In order to avoid losing stored data on cassette or diskette storage mediums, write protecting is a method to stop any unintentional damage.

MICROSPEC

SOFTWARE MEANS BUSINESS FOR THE COMMODORE 64

When it's time to get serious, it's time to boot up MicroSpec business software. Our complete line of business software is made to give you some real applications for your Commodore 64. From data base management to full accounting software, we have the package for you.

It's attention to detail that makes our packages so beautiful and makes them stand out from the rest. We realize that most people are first time users, so we designed all our packages to be completely menu driven and user prompted for each input. We also know that most people use only one disk drive, so we designed all our packages to virtually eliminate disk swapping. Other features like non destructive input routines really make our software easy to use. But all this doesn't restrict you. Pure random access file structure maximizes your disk capacity and allows you to bring up any record for viewing in less than a second.

In our efforts to put together the best packages available, we worked on more than the software. We took the same approach with the documentation as the software. We made it complete and easily understood for the first time user. We even provide sample reports in many cases.

The Demonstration Package, which shows how each program runs, is available for $19.95. So, if you're serious about your 64, call or write for a complete brochure or **go right down to your nearest computer retailer for a demonstration.**

WHEN YOU AND YOUR 64 ARE READY TO GET DOWN TO BUSINESS
GIVE US A CALL

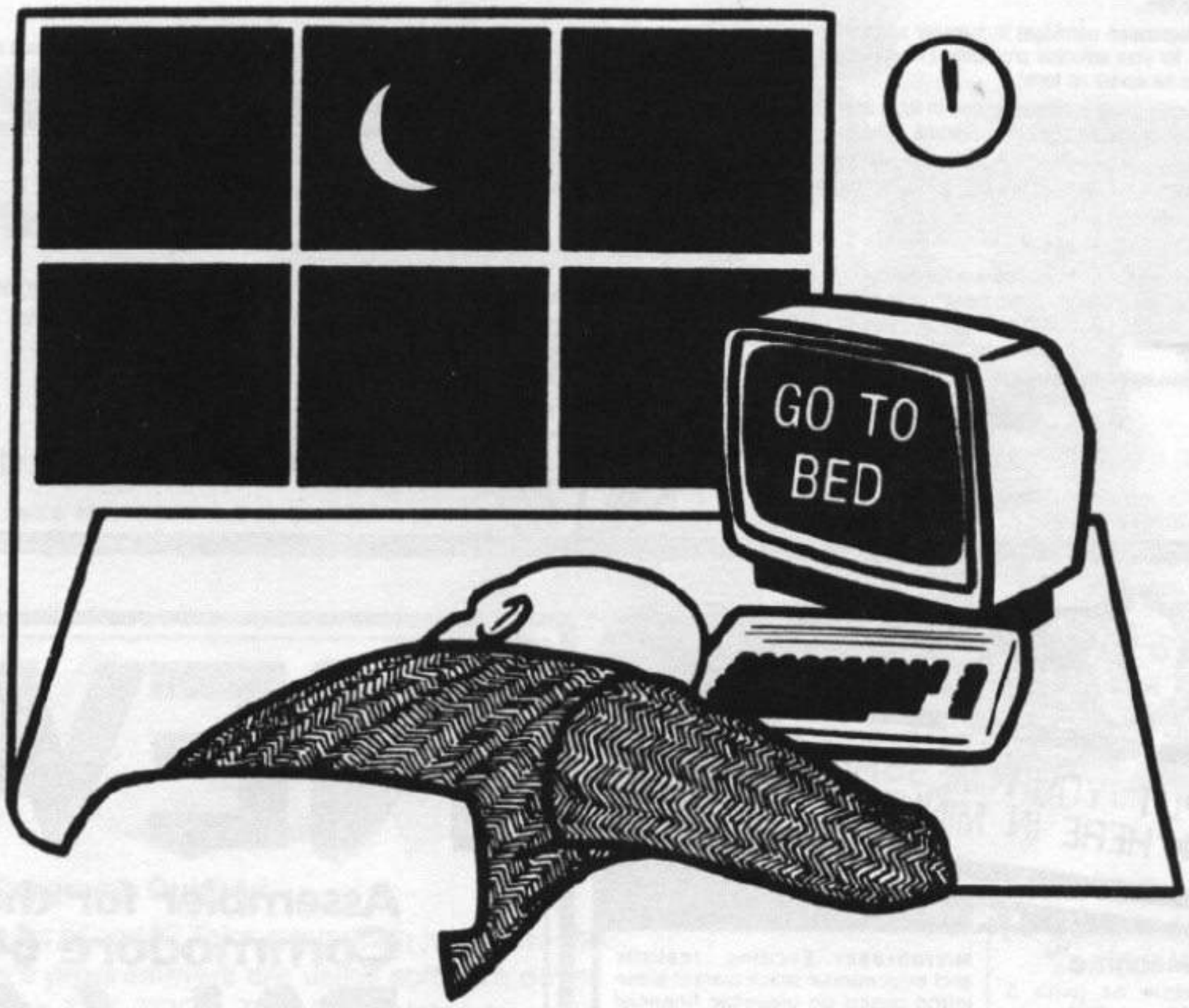
MICROSPEC

P.O. BOX 863085 • PLANO, TX 75086
(214) 867-1333

The MIDNITE SOFTWARE GAZETTE

The PAPER

Five years of service to the PET community.



The Independent U.S. magazine for users of Commodore brand computers.

EDITORS: Jim and Ellen Strasma
Sample issue free on request, from:

635 MAPLE □ MT. ZION, IL 62549 USA

NEW NEW

MPI INTRODUCES SUPER ACTION MEMORY EXPANDER BOARD FOR VIC 20*

Adds 24K and 3 Expansion Slots With Switches and Fuse $109.95

(Expandable To 35K By Simply Adding Memory Chips and Switches)

INTRODUCTORY PRICE

$109.95

Transfer Cartridges To Tape

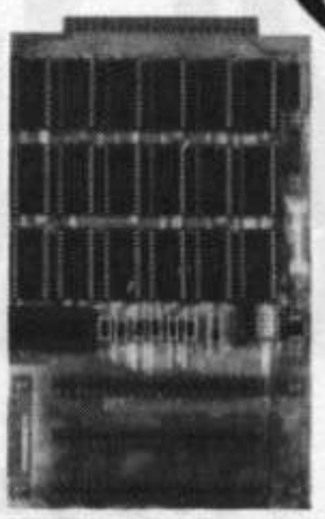
Evade Self Destruct Code With Write Protect Switch

All Boards Factory Tested

Complete Instructions And Illustrations Included.

Fuse Protected

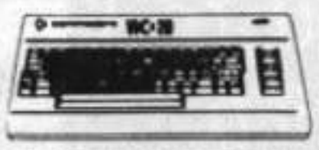
Reset Switch



Pictured Above—
Action Board with all options

24K BOARD FEATURES:

- *Adds 24K Memory (29K with VIC* 5K).
- *Upgrade Board to 35K by adding IC's and switches.
- *Memory switchable in 8K sections. (No need to remove memory board to run your other programs).
- *3 expansion slots with switches (for game or extra utility cartridges).
- *Reset button allows restarting computer without turning power off.
- *.5 amp fuse protected.
- *Switch relocates expansion cartridges in memory so that it can be saved on tape as a backup for your valuable programs. (The unexpanded VIC will not allow cartridges to be saved on tape).
- *Write protect switches allow programs stored in RAM at ROM location or entire Memory to be protected against accidental write.
- *Switch allows memory to be moved between RAM and ROM location. (Useful for developing your own games and saving on tape).
- *Gold plated card edge connector.
- *No other memory expansion needed.
- *Easily plugs into your VIC, no modifications necessary. Saves wear on your VIC 20 since board never needs to be removed or power turned off and on to run other tapes or cartridges.
- **Optional 35K memory (40K with VIC* 5K).



24K memory, 3 expansion slots, switches assembled and tested **$109.95**

Same as above with sockets that allow you to later add your own memory chips to bring memory from 24K to 35K **$124.95**

Full 35K memory, 3 expansion slots, 3K expander mode, eprom socket (switch selectable between BLK 3 & BLK 5) and all switches assembled and tested (eprom not included) **$149.95**

Bare 35K board with complete instruction and parts list **$39.95**

Prices are in US dollars.

COD Orders: 816–444–4651

MIDWEST PERIPHERAL INDUSTRIES

MPI

Add $5 for shipping and handling
Mo. residents add 5½% sales tax
Send check or money order to

MPI
Box 8123-B
Kansas City, MO 64112

Personal checks – Allow 3 weeks to clear.

VIC 20* is a registered trademark of Commodore.

NEW NEW

Reduction of an actual sign

The Banner Machine™

For the Commodore 64 (with 5 fonts) or the VIC-20 with 24K memory. • Use on any Epson MX with Graftrax or the FX and RX printers. • Menu-driven program operates like a word processor. • Makes signs up to 10" tall by any length. • Makes borders of variable width up to ¾". • 8 sizes of letters from ¾" to 6½" high. • Proportional spacing; Automatic centering; Right and left justifying. $49.95 Tape or Disk (Specify computer equipment)

For the Commodore 64:

Space Raider An amazing arcade simulation. Your mission is to destroy the enemy ships. $19.95

Super Roller Challenging dice game. Sprite graphics and sound. Yahtzee-style rules of play. $14.95

Formulator A formula scientific calculator designed for tasks which require repetitive arithmetic computations. You can save formulas and numeric expressions. $39.95

Preschool Educational Programs ABC Fun; 123 Fun; and Ginger the Cat with: Addition and Subtraction, Number Hunt, and Letter Hunt. All programs have bright color, music, and action. Each $14.95

Microbroker Exciting, realistic and educational stock market simulation based on plausible financial events. $34.95 Tape or Disk

Sprite Editor The easy way to create, copy, alter, and save up to 224 sprite shapes. $24.95

Cross Reference Generator for BASIC programs Displays line numbers in which any word of BASIC vocabulary appears. Allows you to change variable name and ask for lines where it appears, and more. $19.95

For the VIC-20:

Caves of Windsor A cave adventure game. The object is to restore wealth and happiness to the small village of Windsor. $14.95

Burger & Fries Fast action joystick game. Eat the burgers and fries but avoid the shakes for a top score. $14.95

Catalog available Dealer inquiries invited

PHONE ORDERS: (703) 491-6502
HOURS: 10 a.m. to 4 p.m. Mon.—Sat.

Cardinal Software™

Distributed by
Virginia Micro Systems
13646 Jeff Davis Hwy
Woodbridge, VA 22191

Commodore 64 and VIC-20 are registered trademarks of Commodore Electronics Ltd.

NEW

Assembler for the Commodore 64

PAL64

- **easy to learn**
- **easy to use**
- **fast**
- **comprehensive manual**

Personal assembly language
by Brad Templeton
also available for the Commodore
4,000 – 8,000 – 9,000 series

$99.95 from your local Commodore dealer.
For your nearest dealer call:

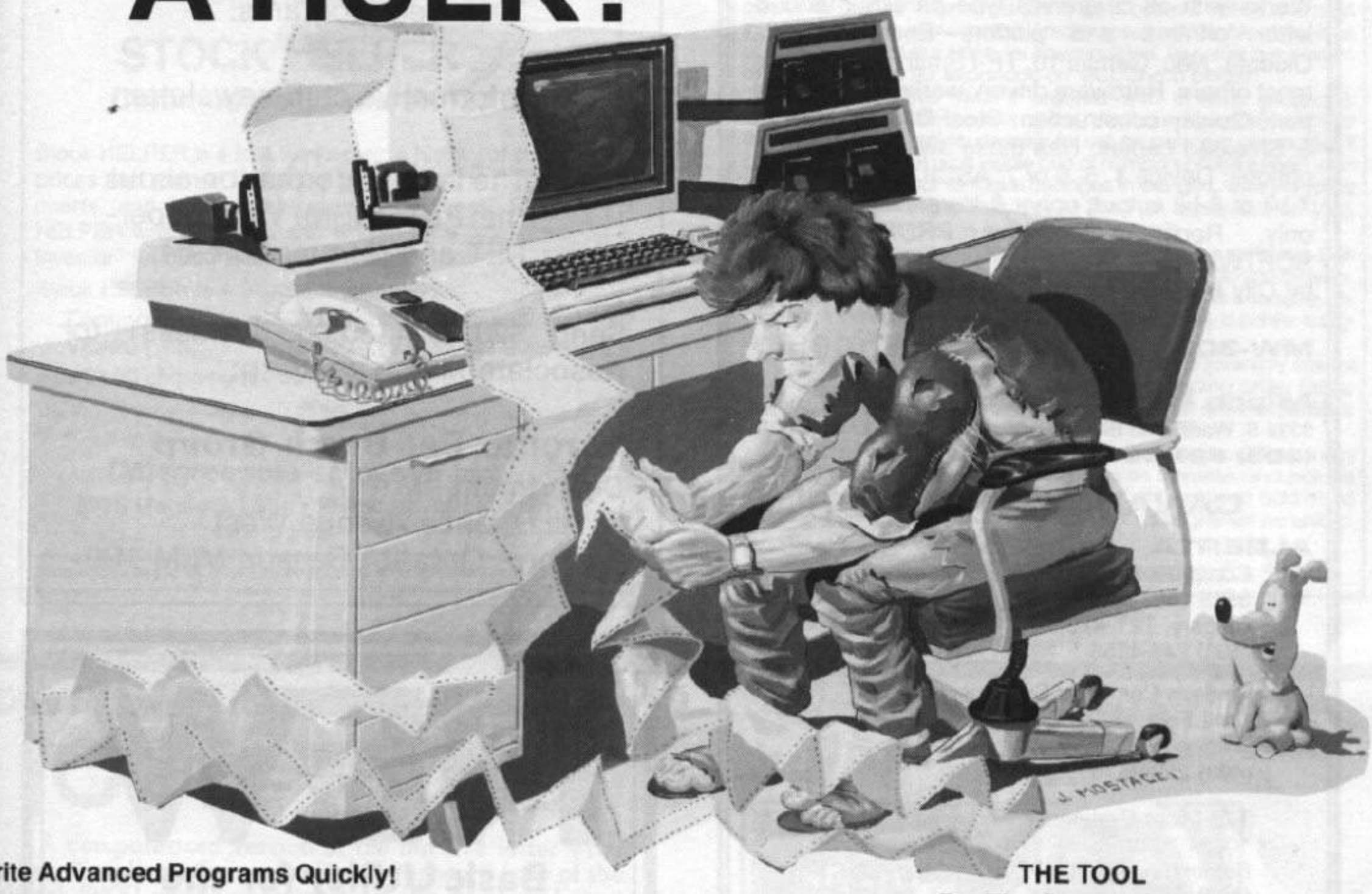
(416) 273-6350

PRO·LINE SOFTWARE

755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO L4Y 4C5

Prices are in US dollars.

IS PROGRAMMING TURNING YOU INTO A HULK?



Write Advanced Programs Quickly!

Tired of writing reams of code? Take a quantum jump into the future! Tommorrow's programmers are using software development tools such as THE TOOL. THE TOOL lets you make use of powerful machine language subroutines. Your programs will execute fast using less code. Input/output routines and professional looking screens are easily created.

Features of **THE TOOL** include :

- Screen Design functions which allow controlled input and output
- High Resolution Graphics with alpha/numeric display
- Screen Save and Load functions (for hi-res and text screens)
- Structured BASIC instructions , e.g. IF THEN ELSE
- Programming Aids (e.g. auto, renumber, delete, find, trace, hardcopy)
- 2 keystroke disk commands (DOS support extensions)
- Game Design Instructions (joy, scroll, screen, colour)
- A 50 page user manual

THE TOOL
For The Commodore 64 ™
$65.00

developed by Micro Application

distributed by BMB Compuscience Canada Ltd.
500 Steeles Avenue,
Milton, Ontario
L9T 3P7
416–876–4741

Dealer Inquiries Invited

Name _____

Address _____

Prov/State _____ Postal/Zip Code _____

Order ☐VISA ☐MasterCard ☐Cheque

Acc# _____ Expiry _____

Please include numbers above name

Add $2.00 for shipping & handling
Ontario residents add 7% sales tax.

MW-302: VIC-20/64

Parallel Printer Interface.

Works with all centronics type parallel matrix & letter printers and plotters—Epson, C.Itoh, Okidata, Nec, Gemini 10, TP-I Smith Corona, and most others. Hardware driven; works off the serial port. Quality construction: Steel DIN connectors & Shielded cables. Has these switch selectable options: Device 4, 5, 6 or 7; ASCII or PET ASCII; 7-bit or 8-bit output; upper & lower case or upper only. Recommended by PROFESSIONAL SOFTWARE for WordPro 3 Plus for the 64, and by City Software for PaperClip.

MW-302 . . . . Canadian $189.95

Micro World Electronix, Inc.
3333 S. Wadsworth Blvd. #C105, Lakewood, CO 80227
(303) 987-2671

CANADIAN DEALERS

ALBERTA

Computer Shop of Calgary
3515 18th St. S.W.
Calgary, T2T 4T9
(403) 243-4356

Hindson Computer Systems, Ltd.
7144 Fisher St. S.E.
Calgary, T2H 0W5
(403) 252-9576

TJB Micro Systems, Ltd.
10991 124th St.
Edmonton, T5M 0H9
(403) 433-3161

BRITISH COLUMBIA

Conti Electronics
7204 Main Street
Vancouver, V5X 3Y4
(604) 324-0505

ONTARIO

MGI Computer Corp.
1501 Carling Ave.
Ottawa, T1Z 7M1
(613) 722-1000

Richvale Telecommunications
10610 Bayview (Bayview Plaza)
Richmond Hill, L4C 3N8
(416)884-4165

SASKATCHEWAN

Micro Shack of West Canada
607 45th St. West
Saskatoon, S7L 5W5
(306) 244-6909

COMMODORE USERS

Join the largest, active Commodore users group in North America and get—

— Access to club library of over 3000 free programs.

— Informative club newsletter.

— The latest information about the PET, CBM, VIC, Super-PET and Commodore-64.

Send $20.00 ($30.00 overseas) for Associate Membership to:

Toronto Pet Users Group
Department "M"
381 Lawrence Avenue West
Toronto, Ontario, Canada M5M 1B9

NEW

Basic Utility for the Commodore 64

POWER 64

•easy to learn
•easy to use
•program faster and more efficiently with better results
•MOREPOWER included FREE

Powerful Programmer's Utility
by Brad Templeton
Manual by Jim Butterfield

$99.95 from your local Commodore dealer.
For your nearest dealer call:
(416) 273-6350

PRO·LINE SOFTWARE
755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO L4Y 4C5

STOCK HELPER™

Commodore 64 and VIC-20

Stock HELPER is a tool to maintain a history of stock prices and market indicators on diskette, to display charts, and to calculate moving averages. Stock HELPER was designed and written by a "weekend investor" for other weekend investors.

Stock HELPER is available on diskette for:

Commodore 64	$30.00	($37.00 Canadian)
VIC-20 (16K)	$27.00	($33.25 Canadian)

plus $1.25 shipping ($1.55 Canadian)

The VIC-20 version only charts 26 bi-weekly periods rather than 52 weekly periods.

(M)agreeable software, inc.
5925 Magnolia Lane • Plymouth, MN 55442
(612) 559-1108

(M)agreeable and HELPER are trademarks of (M)agreeable software, inc.
Commodore 64 and VIC-20 are trademarks of Commodore Electronics Ltd.

Intelligent Software for the Commodore computers

Catalog 5/1/83

My line of programs (such as it is) consists of the following products. All are written for Commodore computers; any of my programs will load and run without modification in the entire line (including older PET's).

1. Word Processor; $25. It includes the following features: VERY fast file routines, including a disk file catalog; automatic form handling on tractor- or friction-feed printers; fully imbedded margin, justification, spacing, formatting, and paging controls; block commands and error-trapping in editing mode; and a spool routine (formatted output to disk for later mass printing). I believe W/P is the most thoroughly tested, user-oriented word processor available at this time at anywhere near the price, for any machine. Requires a minimum of 10k of memory (8k expansion on VIC), and a printer.

2. Copycalc; $20 ($15 if ordered with another program). Copycalc is a simplified version of the "electronic spreadsheets" that are becoming extremely popular for use on personal computers. It allows the user to set up a visible grid of numbers on the screen, and use the screen-editor to make changes in the grid, with the totals reflecting the changes. Requires 6k RAM (3k expansion on VIC); smaller version available for unexpanded VIC.

3. Baseball Manager; $30. This program maintains complete batting statistics for a baseball or softball league of up to 250 players. It generates reports on a player, team, or the entire league (including standings). It requires a minimum 10k of RAM; a printer is suggested but not required.

4. Inventory; $30. A general-purpose perpetual inventory control program. It produces a variety of reports, including order forms; multiple vendors are supported. Requires 10k of RAM; a printer is suggested.

All programs: support cassette and disk files and the CBM printers (easily modifiable to other printers), come on cassette, and include documentation. Prices include shipping; Calif. residents add 6%. All programs are copyrighted by the author; those rights will be enforced. Programs available from:

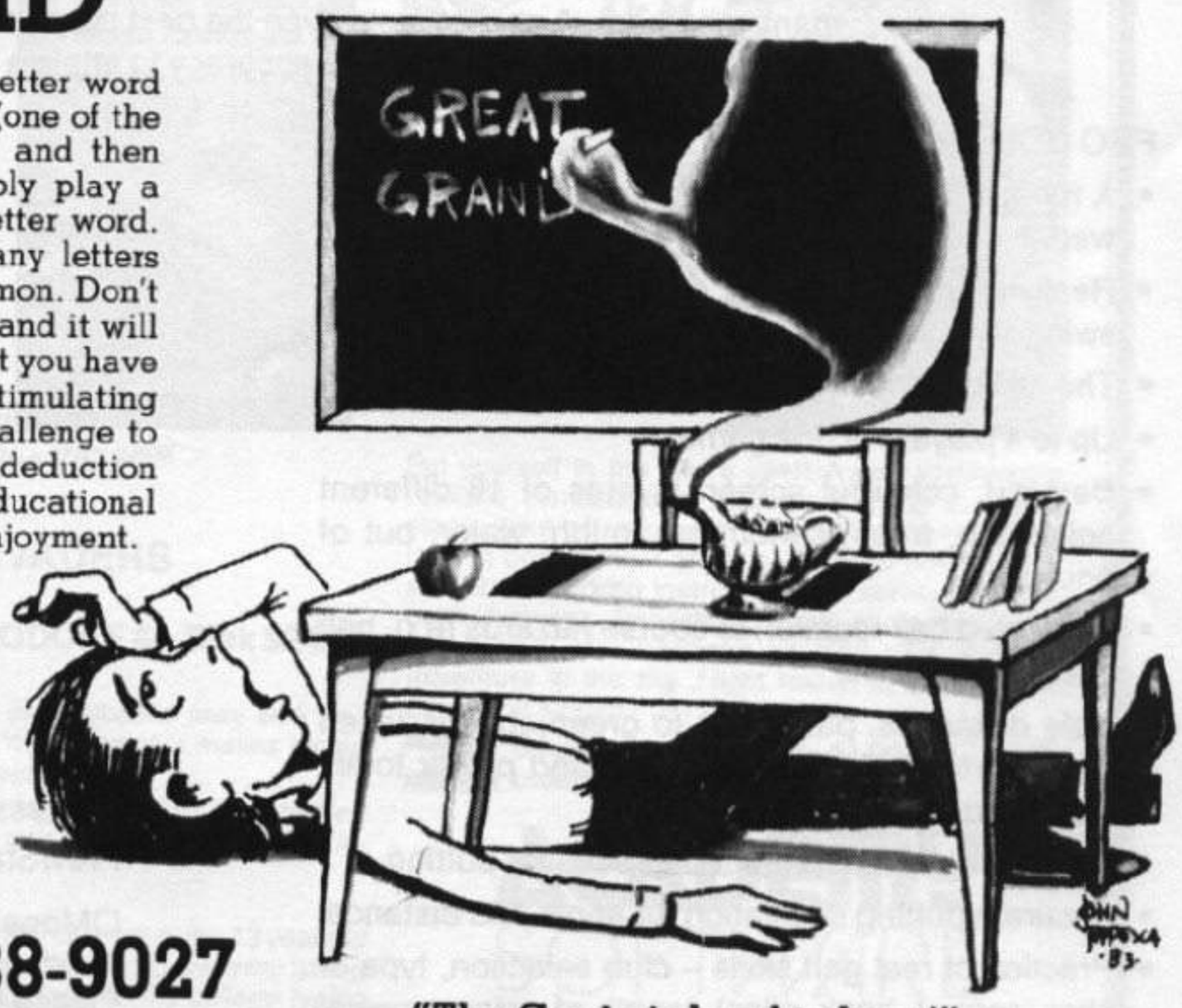
William Robbins, Box 3745, San Rafael, CA 94912

Disk Software for the Commodore 64™

JOT-A-WORD™

A computerized version of the old five letter word game. Simply pick a secret five letter word (one of the almost 5000 words contained on the disk) and then play against the Jot-A-Word Genie or simply play a solitaire version. Start by typing in a five letter word. The Genie responds by telling you how many letters your guess and the secret work have in common. Don't try to cheat, because the Genie is too smart and it will not accept non-words or continue a game that you have given it wrong scores. This is a simple but stimulating game for ages 9 to senior citizen. A real challenge to your intellect, reasoning powers, logic and deduction skills. It's simply hard to beat; as a fun and educational experience! Graphics and music add to the enjoyment.

MasterCard VISA



"The Genie is hard to beat!"

ONLY $29.95

MICRO WARE

1342B RT. 23
BUTLER, N.J. 07405

Dealers & Distributors Inquiries Invited

201-838-9027

Prices are in US dollars.

PRO GOLF

Here is your chance to play golf on a championship course without all the headaches of getting a tee time, waiting for that slow foursome ahead of you, losing balls, getting rained out or spoiling a good handicap. This game may be played in the privacy of your home or in a clubhouse lounge for the enjoyment of many members. A challenge to even the best players, this game requires a high degree of practice, expertise and accuracy to attain a good score.

PRO GOLF Features :

- A full range of golf clubs (driveway, fairway wood, wedge and irons 2–9)
- Realistic shot distances depending on club and swing
- The ability to hook or slice a shot
- Up to 4 players in one game
- Detailed, colourful screen layouts of 18 different holes (tee, trees, sandtraps, rough, water, out of bounds)
- Simulated ball reaction to course hazards (e.g. ball bounces off trees)
- Hole distances, par, yards to green, strokes taken on hole, total strokes per round and player totals displayed
- A full screen enlargement of greens for putting
- Accurate putting simulation for angle and distance
- Practice of real golf skills – club selection, type of shot (normal, hook, slice), length of swing, special shot strategy (e.g. chipping, getting around or over trees, water, sandtraps)

PRO GOLF
For The Commodore 64
$34.95

written by George Adams

distributed by BMB Compuscience Canada Ltd.
500 Steeles Avenue,
Milton, Ontario
L9T 3P7
416–876–4741

Dealer Inquiries Invited

Name _____

Address _____

Prov/State _____ Postal/Zip Code _____

☐Money Order ☐VISA ☐MasterCard ☐Cheque

Acc# _____ Expiry _____

Please include numbers above name

Add $2.00 for shipping & handling
Ontario residents add 7% sales tax.

Let The SMART 64 Terminal

COMMODORE 64* Do The DRIVING

No matter which direction you wish to travel in, experience the advantage of computer communications with The SMART 64 Terminal. Discover the program that puts you on the Right Road to: Public-Access Networks, University Systems, Private Company Computers and Financial Services.

The SMART 64 Terminal designed with Quality-Bred features, Affordable Pricing . . . And Service.

So why not travel the communications highways the SMART way!

Suggested $39.95 Retail

Accessories included:

- ☐ *Selective Storage of Received Data.*
- ☐ *Alarm Timer.*
- ☐ *40 or 80 Col. Operation*.*
- ☐ *Auto-Dial.*
- ☐ *User-Defined Function Keys, Screen Colors, Printer and Modem Setting.*
- ☐ *Screen Print.*
- ☐ *Disk Wedge Built-In!*
- ☐ *Formatted Lines.*
- ☐ *Review, Rearrange, Print Files.*
- ☐ *Sends/Receives Programs and Files of ANY SIZE.*
- ☐ *Adjustable transmit/receive tables allow custom requirements. These and other features make The SMART 64 Terminal the best choice for grand touring telecommunications.*

*Commodore 64 registered trademark of Commodore Business Machines Inc.
*Supports 80-column cartridge by Data 20 Corporation.

Dealer Availability
Call (203) 389-8383

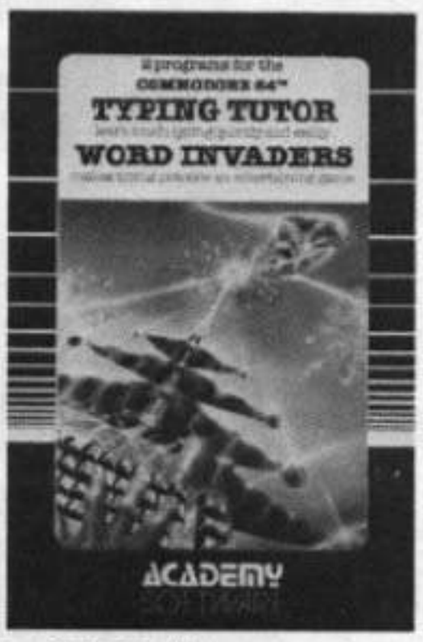
MICROTECHNIC® SOLUTIONS
P.O. BOX 2940, New Haven, Ct. 06515

Prices are in US dollars.

VIC 20™ COMMODORE 64™

JOIN THE COMPUTER REVOLUTION WITH A MASTERY OF THE KEYBOARD!

In the age of the computer, *everyone* from the school child to the Chairman of the Board should be at home at the computer keyboard. Soon there will be a computer terminal on every desk and in every home. Learn how to use it right ...and have some fun at the same time!



Rated THE BEST educational program for the VIC 20™ by Creative Computing Magazine

TYPING TUTOR PLUS WORD INVADERS

The proven way to learn touch typing.

COMMODORE 64 Tape $21.95 COMMODORE 64 Disk $24.95
VIC 20 (unexpanded) Tape $21.95

Typing Tutor plus Word Invaders makes learning the keyboard easy and fun! Typing Tutor teaches the keyboard in easy steps. Word Invaders makes typing practice an entertaining game. Highly praised by customers:

"Typing Tutor is great!", "Fantastic", "Excellent", High quality", "Our children (ages 7-15) literally wait in line to use it.", "Even my little sister likes it", "Word Invaders is sensational!"

Customer comment says it all . . .

". . . it was everything you advertised it would be. In three weeks, my 13 year old son, who had never typed before, was typing 35 w.p.m. I had improved my typing speed 15 w.p.m. and my husband was able to keep up with his college typing class by practicing at home."

NEW!

IFR (FLIGHT SIMULATOR) CARTRIDGE FOR THE VIC 20

$39.95
JOYSTICK REQUIRED

Put yourself in the pilot's seat! A very challenging realistic simulation of instrument flying in a light plane. Take off, navigate over difficult terrain, and land at one of the 4 airports. Artificial horizon, ILS, and other working instruments on screen. Full aircraft features. Realistic aircraft performance — stalls/spins, etc. Transport yourself to a real-time adventure in the sky. Flight tested by professional pilots and judged "terrific"!

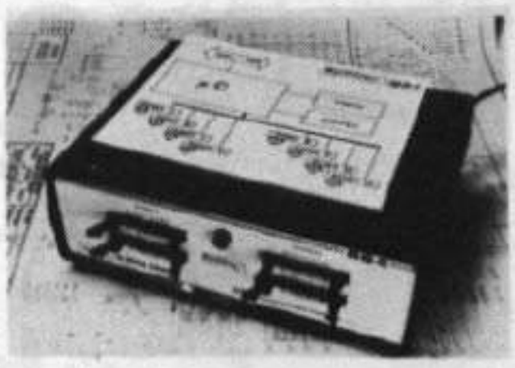
VISA Shipping and handling $1.00 per order. CA residents add 6% tax. MasterCard

ACADEMY SOFTWARE
P.O. Box 9403, San Rafael, CA 94912 (415) 499-0850

Programmers: Write to our New Program Manager concerning any exceptional VIC 20™ or Commodore 64™ game or other program you have developed.

BUSSter™

REAL-WORLD INTERFACES FOR COMPUTERS



ANALOG AND DIGITAL INPUT/OUTPUT

The BUSSter line of analog and digital products was designed to collect data and to output signals to laboratory and industrial equipment in conjunction with a microcomputer system. These powerful self-contained modules reduce a computer's workload by providing read or write operations to external devices. They are controlled as slave interfaces to real-world physical applications. Control is over an IEEE-488 (GPIB) bus or RS-232 port. The internal buffer and timer provide flexibility by allowing the BUSSter to collect data while the host computer is busy with other tasks. All units have a timer which operates from .01 seconds to 46 hours.

- **BUSSter** — 64 channel digital input module to read 64 digital signals. Built-in buffer *$495.00*
- **BUSSter B64** — 64 channel digital output module to send 64 digital signals ... *$495.00*
- **BUSSter C64** — 64 channel digital input/output module to read 32 and write 32 digital signals. Built-in buffer *$495.00*
- **BUSSter D16** — 16 channel analog input module to read up to 16 analog signals with 8 bit resolution (¼%). Built-in buffer ... *$495.00*
- **BUSSter D32** — 32 channel version of the D16 ..... *$595.00*
- **BUSSter E4** — 4 channel analog output module to send 4 analog signals with 12 bit resolution (.06%) *$495.00*
- **BUSSter E8** — 8 channel version of the E4 ..... *$595.00*
- **BUSSter E16** — 16 channel version of the E4 ..... *$695.00*

Add the suffix -G for IEEE-488 (GPIB) or -R for RS-232.

All prices are USA only. Prices and specifications subject to change without notice.

30 DAY TRIAL —
Purchase a BUSSter product, use it and if you are not completely satisfied, return it within 30 days and receive a full refund.

US Dollars Quoted—$10 Shipping & Handling
MASTERCARD / VISA

CmC

Connecticut microComputer, Inc.
INSTRUMENT DIVISION

36 Del Mar Drive • Brookfield, Ct. 06804
(203) 775-4595 TWX: 710-456-0052

CURSOR

For your Commodore 64

For only $12.95 each, our **CURSOR 64** tapes are your best buy for the Commodore 64. They take advantage of the color, sound, and sprites that make the 64 such a delight to use. Most of our packages include three excellent Basic programs on one cassette tape. The programs are not copy protected, so you can look at the source code, and learn how to make the 64 do its tricks.

We don't have room to describe all 25 of our **CURSOR 64** programs here. As a sample, you may want to order tape 64-5 with the exciting **Godzilla** program. You'll be challenged as you try to save Tokyo from from the rampaging Godzilla. Or try tape 64-3 with the popular **Miser** text adventure that will take you hours to solve (even if you cheat and read the program source).

We have super programs for the VIC 20, such as **Dungeon** ($12.95), a visual adventure for 16K VICs. Our **VIXEL** programs are also popular with VIC owners. And, we still sell all 30 of the original **CURSOR** cassettes for the original PET and CBM.

Call or write for a catalog today. Be sure and tell us whether you have a 64, a VIC, or a PET. We welcome credit cards, and ship most orders the same day they are received. Dealer inquiries invited.

CURSOR 64, Box 6905
Santa Barbara, CA 93110
805-683-1585

MAILPRO
STEVE PUNTER'S DATA ORGANIZER AND MAILING LIST PROGRAM FOR COMMODORE 64™

COMPARE THESE FEATURES:

- **fast file definition**
- **easy updating**
- **rapid printing with total format and record selection control**
- **WORDPRO compatible**
- **up to 4000 records on 1541**

MAILPRO 64.....$129.95

Also available for COMMODORE 8032 . . . $179.95

Call for the name of your local dealer:

PRO·LINE SOFTWARE

PRO-LINE SOFTWARE LTD.
(416) 273-6350
755 THE QUEENSWAY EAST, UNIT 8
MISSISSAUGA, ONTARIO CANADA, L4Y 4C5

Finally! An Affordable Full-Size, Full-Feature PRINTER

For your **VIC-20®, C-64®**
ATARI®
Centronics Parallel Types
And RS-232 Serial Types

SUG. LIST $299

$229.95!!

BASIC PRINTER
(Requires one
Option Below)

FEATURES:

- Full graphics capability.
- In the graphic mode, a column of graphic data can be repeated as many times as you want with a single command.
- Double width character output under software control (5 char. per inch).
- Print position addressable by character or dot (positioning control).
- Graphic character and double width character modes can be intermixed on a single line.
- Automatic printing. When the text exceeds the maximum line length no data is lost due to overflow.
- Self-test printing mode.
- Paper width is adjustable up to 10 inches. Standard plain paper.
- 50 cps print speed.
- 80 characters per line.
- 5 × 7 dot matrix.
- Full 2 yr. Warranty.
- Foreign character sets
  For U.S., U.K., Sweden, and Germany.

Any of these **Options** allow you to connect and print - cables included.

APROPRINT-2064™ (pictured) . . . .Add: $35.95
For Commodore **VIC-20 & C-64** - Cable included.

APROPRINT-4080™ . . . . . .Add: $45.95
For **all Atari Computers** - Cable included.

APROPRINT-1000™ . . . . . .Add: $29.95
RS-232-Serial - Name your computer

APROPRINT-8000™ . . . . . .Add: $29.95
Centronics type Parallel - Name your computer

ADD: $8.00 shipping (cont. USA), $25.00 (Canada, HI, AK)

(All other foreign orders Add $55.00 (shipped by Air)

The ONE VIC-20® Memory Expansion Board that DOES IT ALL!

Maximum Memory allows you to use more powerful programs for:
- EDUCATION • ENTERTAINMENT • MAIL LISTS
- BUSINESS APPLICATIONS • FINANCIAL RECORDS

APROPOS TECHNOLOGY
RAMAX™

A perfect investment to give your family and yourself more enjoyment and use from your home computer! The ease of operation, the neat appearance, and the real POWER it adds to your VIC at this low price makes it a MUST for every VIC home!

SPECIAL LOW PRICE!

Only $149.00

Price includes shipping and handling within Continental USA. Foreign orders please add $15.00. Calif. Residents add 6% sales tax.

10 DAY MONEY-BACK GUARANTEE
If not satisfied, simply return in original condition for your money back.

RAMAX Jr.™

Already own an 8k Expander? Get the NEW **RAMAX Jr.™**! Identical to the RAMAX™ except with 19k instead of 27k. Our instructions will show you how to use your 8k as BLK 3 with Jr. to get the full complement of memory!

Special Only $129.00
Shipping included

To equal the total memory of RAMAX™ you would have to buy a 16k Memory Expansion, PLUS an 8k Expansion, PLUS 3k Expansion. THEN you would need a "mother board". With RAMAX™ you buy just ONE piece . . . at ABOUT HALF THE PRICE!

RAMAX™ Features and Specifications:

- Adds up to a full 27k bytes of additional RAM to the standard VIC-20's internal RAM of 5k.
- Built-in switch allows User selection of any combination of 5 areas of RAM memory*:
  BLK1 (8k: Adr. 8192-16383
  BLK 2 (8k: Adr. 16384-24575)
  BLK 3 (8k: Adr. 24576-32767)
  BLK 5 (Adr. 40960-49151, allows/disallows 8k ROM games)
  RAM (3k: Adr. 1024-4095)
  RESET (Resets computer without power off/on)
- Built-in electrical Fuse to protect equipment.
- Totally self-contained. No external power supply needed.
- Two (2) extension connectors allow ANY additional cartridges and/or devices designed for the VIC expansion port.
- Very low power consumption (.175 amp usual).
- High reliability gold-plated connectors are designed for long life.
- Complete Operating Manual.
- 6 month parts and labor warranty to original purchaser.
- Factory service.

*Many VIC-20 cartridges and programs require certain configurations of the memory (i.e certain games will only run on the unexpanded VIC while others require the upper portion of the expanded memory). With RAMAX™ you have switches that turn-on and turn-off portions of the memory to provide the right area of memory - all without plugging or unplugging. It's so easy!

TO ORDER:

Send Check or Money Order For the Total
Calif. residents add 6% tax.
Or Contact your Local Dealer
Phone orders Call **(805) 482-3604**

VISA All Prices U.S. Dollars MasterCard

DEALER INQUIRIES WELCOME

WE SERVICE WHAT WE SELL

VIC-20 & Commodore-64 are registered trademarks of Commodore International. Atari is a trademark of Atari Inc.

APROPOS TECHNOLOGY
1071-A Avenida Acaso
Camarillo, CA 93010

New Product!

APROSPAND-64™ Gives your Commodore 64 full expandability. This superbly designed expansion module plugs into the 64 & gives you 4 switchable (singly or in any combination) expansion connectors - plus fuse protection - plus a reset button! only **$54.95**
Shipping included

APROPOS TECHNOLOGY

MICROCOMPUTER SUPPLIES

100% Guaranteed

DISKETTES FROM
10/$25.00
S.S. S.D

Shop Without Going Shopping

MEMOREX		DYSAN	
3481 SS/DD	$33.00 per 10	104/1D SS/DD Soft	$54.00 per 10
3491 DS/DD	$45.00 per 10	104/2D DS/DD Soft	$76.00 per 10
3481-2 SS/DD 2 Pack Mailer	$8.00 plus 75¢ postage		

All other sizes and format available and volume discount.

Flip 'N' File, hold 50, 5¼" diskettes	$38.50 each
Flip Pac, holds 10 5¼" diskettes	$ 7.00 each
Disk Banks, hold 10 5¼" diskettes	$ 7.00
Plastic Library Cases, choice of colors;	$4.50 each
EPSON RX80 Printer	$525.00
EPSON FX80 Printer	$850.00
EPSON FX100	$CALL
QUADRAM Interfaces	$CALL
COMREX CRII Daisy Wheel Printer	$869.00
COMREX Monitors, 8 models	from $198.00
MX80 Ribbons	$14.50

WRITE OR CALL FOR FREE CATALOG

To order: Send money order, certified cheque, personal cheques must clear our bank, VISA or MASTERCARD. (include card # and expiry date & signature) Add 5% for shipping and handling. Minimum $3.00 per order. Quebec residents add 9% P.S.T.

INTERNATIONAL MARKETING SERVICES

P.O. Box 522, Boucherville, Quebec, J4B 6Y2
(514) 655-9232

FREE CATALOG!

UTILITIES, GAMES, AMATEUR RADIO FOR THE VIC 20 AND COMMODORE 64

NEW ITEM...

C64 WORLD CLOCK - graphic map showing time in cities across the world.....
Tape $ 7.95
Disk $10.95
(+ $2.00 shipping - US. Currency)

Over 30 other programs!

Low prices! Excellent service!

RAK Electronics
PO. Box 1585
Orange Park, FL
32067–1585

Open your mind

© 1983 PATRICK J. WILLIAMS

Reveal secrets of the mind.
Use your Commodore 64 system to analyze yourself, your spouse, your date, relatives and friends. Discover your personality type, career potential, behavior tendencies, values, and the people with whom you will be most compatible. This program requires the use of a "joystick".
Price $32.95 Disk ($27.95 Cassette).

Behavior Modification.
Use your Commodore 64 system to change your behavior patterns through computer hypnosis. Discover how to communicate with *yourself*, on a conscious and subconscious level. Program your own post-hypnotic suggestions. The PSI Biofeedback Device is included with this program.
Price $87.95 Disk ($79.95 Cassette).

Get this software at your local dealer or order direct from:

PSYCOM SOFTWARE INTERNATIONAL

2118 Forest Lake Drive
Cincinnati, Ohio 45244 USA
Telephone: 513 474-2188

This Space Could Be Transacting For You!

Kelly M. George
Advertising Manager
416 876 4741

TELSTAR 64

Sophisticated Terminal Communications Cartridge for the 64.

PFO 10D 00D CP D1 D2 BELL 12:30:00 10:14:36
(TELSTAR's Status Line)

Don't settle for less than the best!

- Upload/Download to/from disk or tape.
- Automatic File Translation.
- Communicates in Industry Standard ASCII.
- Real-Time Clock plus Alarm Clock.
- Line editing capability allows correcting and resending long command lines.
- 9 Quick Read functions.
- Menu-driven.
- Similar to our famous STCP Terminal package.
- Works with Commodore Modems and supports auto-dialing.

The best feature is the price — **only $49.95** (Cartridge and Manual)

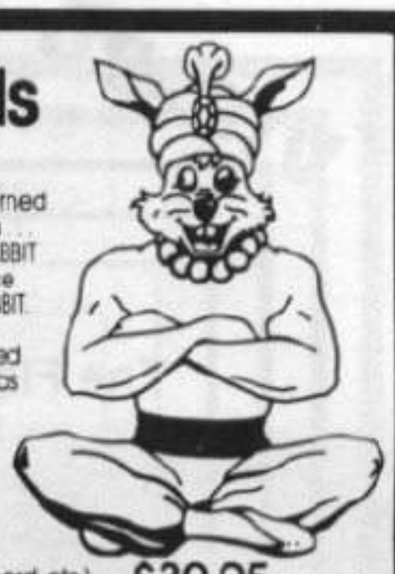
Machine Language Monitor Cartridge
for the CBM 64

More than 20 commands allow you to access the CBM 64's Microprocessors Registers and Memory Contents. Commands include assemble, disassemble, registers, memory, transfer, compare, plus many more.

Someday every CBM 64 owner will need a monitor such as this.

Cartridge and Manual — **$24.95**

8K in 30 Seconds
for your VIC 20 or CBM 64



If you own a VIC 20 or a CBM 64 and have been concerned about the high cost of a disk to store your programs on . . . worry yourself no longer. Now there's the RABBIT. The RABBIT comes in a cartridge, and at a much, much lower price than the average disk. And speed . . . this is one fast RABBIT. With the RABBIT you can load and store on your CBM datasette an 8K program in almost 30 seconds, compared to the current 3 minutes of a VIC 20 or CBM 64, almost as fast as the 1541 disk drive.

The RABBIT is easy to install, allows one to Append Basic Programs, works with or without Expansion Memory, and provides two data file modes. The RABBIT is not only fast but reliable.

(The Rabbit for the VIC 20 contains an expansion connector so you can simultaneously use your memory board, etc.)

$39.95

MAE
NOW THE BEST FOR LESS!

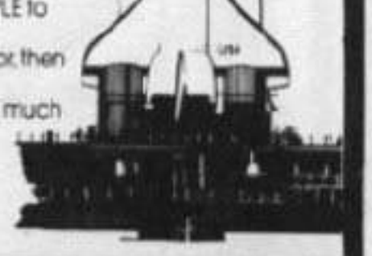
$59.95

For CBM 64, PET, APPLE, and ATARI

Now, you can have the same professionally designed Macro Assembler/Editor as used on Space Shuttle projects.

- Designed to improve Programmer Productivity.
- Similar syntax and commands - No need to relearn peculiar syntaxes and commands when you go from PET to APPLE to ATARI.
- Coresident Assembler/Editor - No need to load the Editor, then the Assembler, then the Editor, etc.
- Also includes Word Processor, Relocating Loader, and much more.
- Powerful Editor, Macros, Conditional and Interactive Assembly, and Auto - zero page addressing.

Still not convinced: send for our free spec sheet!



Eastern House

3239 Linda Dr.
Winston-Salem, N.C. 27106
(919) 924-2889 (919) 748-8446
Send for free catalog!

VISA
MasterCard

Prices are in US dollars.

C 64 PROVINCIAL PAYROLL

A complete Canadian Payroll System for Small Business.

• 50 Employees per disk (1541) • Calculate and Print Journals • Print Cheques • Calculate submissions summary for Revenue Canada • Accumulates data and prints T-4s • Also available for 4032 and 8032 Commodore Computers.

Available from your Commodore Dealer.

Distributed by:

MICROCOMPUTER SOLUTIONS
1262 DON MILLS RD. STE. 4
DON MILLS, ONTARIO M3B 2W7
TEL: (416) 447-4811

TYPRO
DATA MANAGER & WORD PROCESSOR
For COMMODORE 8032 Computer — 8050/4040 Dr.

NOW AVAILABLE FOR THE COMMODORE 64®

DATA MANAGER

Number of records is only limited by your disk capacity. Up to 50 fields per record. Maximum of 75 characters per field. User formatted. Screen editing. Sort and search feature. Pattern match search. Selective field printing and formatting. Form letter addressing. Mailing list and mailing label printing. Format for fanfold Rolodex and index card printing.

WORD PROCESSOR

Screen editing. Automatic line length set. Add, move or delete text. Global edit. Page numbering and titling. Form letter addressing. File append for printing.

BOTH PROGRAMS ABOVE, ONLY $89.00

* Commodore 64 is a trademark of Commodore Electronics, Ltd.

Also for Commodore 64 and 8032
AMORTIZATION SCHEDULE — $30.00 (Disk)
INVENTORY MANAGEMENT — $55.00 (Disk)

All software is fully supported for updates and revisions for up to six months after purchase.

Specify Computer model number and Disk model number.

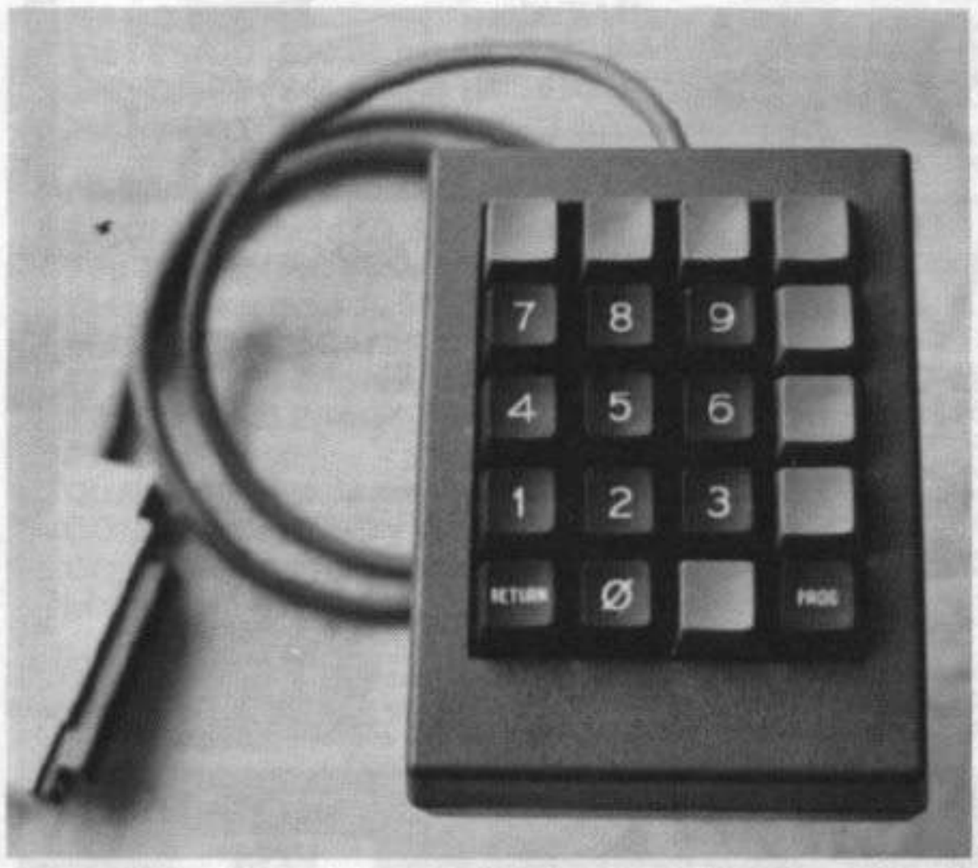
INPUT SYSTEMS, INC.
25101 S.W. 194 Ave. Homestead, FL 33031 (305) 245-3141
DEALER INQUIRIES INVITED.

GOSUB

VIC-20 C64

INTERNATIONAL INCORPORATED

The Flexikey System



Retail
$69.95

Features:

19 Keys, each of which may have 3 seperate definitions!

Complete documentation including program listings!

Works on the VIC20 (Expanded) and C-64 computers!

Compatible with most existing software!

Great for use with business programs and electronic spread sheets!

Ideal for machine language programmer!

VISA & MASTERCARD WELCOME

Prices subject to change

Dealer Inquires Invited - **(316) 265-9858**
GOSUB International - 501 E. Pawnee - Suite 430
Wichita, Kansas 67211

*C-64 and VIC 20 are registered trademarks of Commodore International.

We'll back you up!

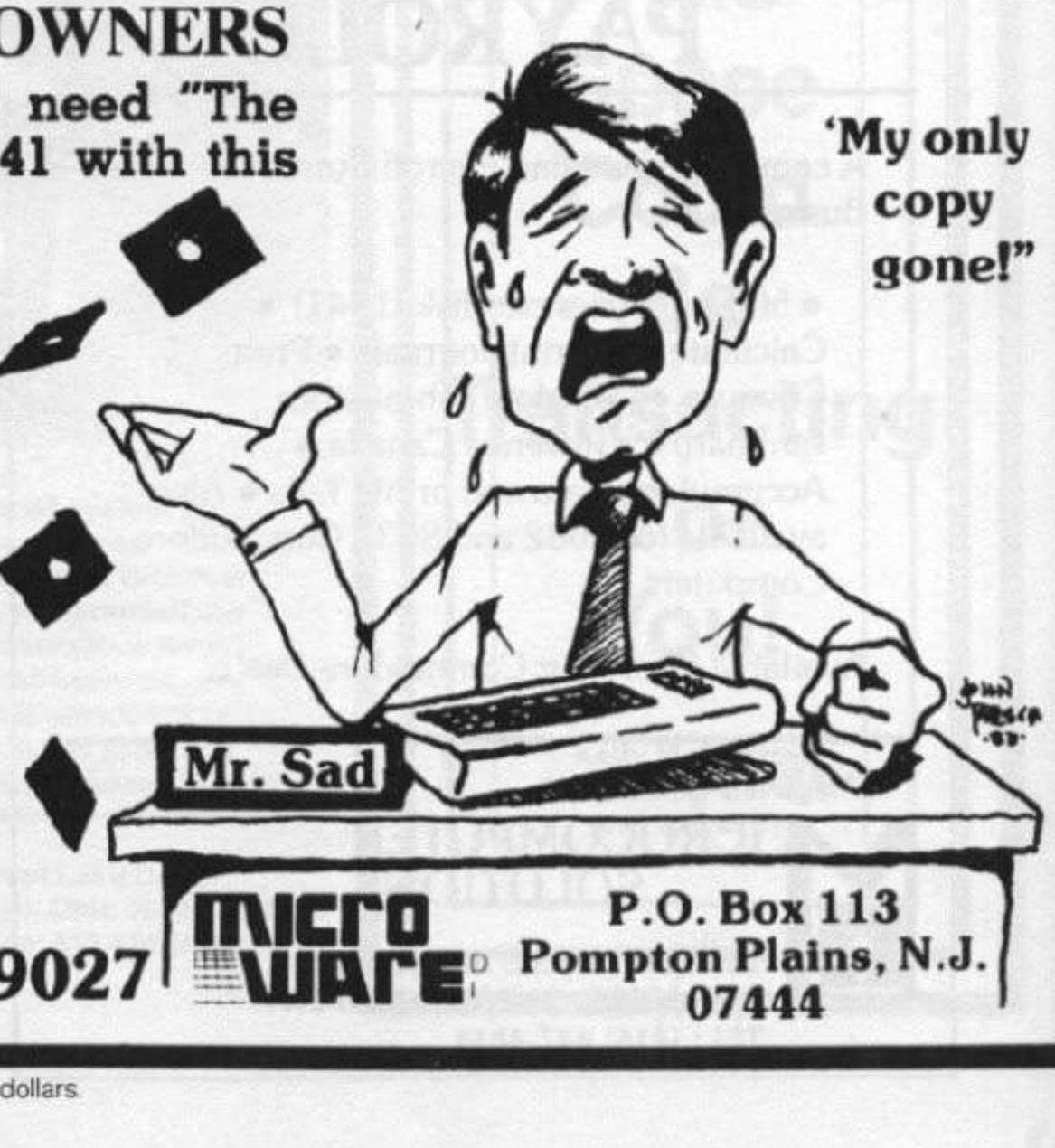
MasterCard VISA

ATTENTION COMMODORE 64 OWNERS

If you own a disk drive then you'll need "The Clone Machine". Take control of your 1541 with this package that includes:

1.) Complete and thorough users manual
2.) Copy with one or two drives
3.) Investigate and back-up many "PROTECTED" disks
4.) Copy all file types including relative types
5.) Edit and view track/block in Hex or ASCII
6.) Display full contents of directory and print
7.) Change program names, add, delete files with single keystroke
8.) Easy disk initialization
9.) Supports up to four drives

~~List $44.95~~ **Special intro $39.95**



Dealers & Distributors Inquiries Invited

CALL (201) 838-9027

MICRO WARE

**P.O. Box 113
Pompton Plains, N.J.
07444**

Prices are in US dollars

Commodore 64
and
VIC-20

SuperTerm

$99.95

Telecommunications with a difference!

Unexcelled communications power and compatibility, especially for professionals and serious computer users. Look us over; **SuperTerm** isn't just "another" terminal program. Like our famous Terminal-40, **it's the one others will be judged by.**

- **UP/DOWNLOAD FORMATS** –CBM, Xon-Xoff, ACK-NAK, etc.
- **DISPLAY MODES** — 40 column; 80/132 with side-scrolling
- **EMULATION** — 42 popular terminal protocols
- **FUNCTION KEYS** — 8 standard, 52 user-defined
- **BUFFERS** — Receive, Transmit, Program, and Screen
- **EDITING** — Full-screen editing of Receive buffer
- **FILE CONVERSION** — ASCII to PGM, PGM to ASCII
- **PRINTING** — Continuous printing with Smart ASCII and parallel printer; buffer printing with other interfaces or VIC printer
- **DISK SUPPORT** — Directory, Copy, Rename, Scratch, etc.
- **FLEXIBILITY** — Select baud, duplex, parity, stopbits, etc.

Program options are selected by menus and function keys. For maximum convenience, an EXEC file sets options on start-up. SuperTerm may be backed-up for safety. Software on disk or cassette, with special cartridge module.

Write for the full story on SuperTerm; or, if you already want that difference, order today!

Requires: Commodore 64 or VIC-20, disk drive or Datasette, and compatible modem. VIC version requires 16K memory expansion. Please specify VIC or 64 when ordering.

Just need UP/DOWNLOAD?

If you don't yet need SuperTerm's power, perhaps **Terminal-40 Plus** (VIC) or **'64 Terminal Plus** is right for you. We took our top-rated, smooth-scrolling terminal programs, added up/download, disk commands, and even more convenience. Then we put them on disk for fast loading, just like you wanted. Need we say more?

Only $49.95 (VIC version requires 8K mem exp)

PS. Trade in your original Terminal-40 or '64 Terminal and deduct $10.00.

VIC 20 and Commodore 64 are trademarks of Commodore Electronics, Ltd.

(816) 333-7200

Send for a free brochure.

MIDWEST MICRO inc.

MAIL ORDER: Add $1.50 shipping and handling ($3.50 for C.O.D.); VISA/Mastercard accepted (card# and exp. date). MO residents add 5.625% sales tax. Foreign orders payable U.S.$, U.S. Bank ONLY; add $5 shp/hndlg.

311 WEST 72nd ST. • KANSAS CITY • MO • 64114

Prices are in US dollars.

This Space Could Be Transacting For You!

PRO·LINE SOFTWARE

A CANADIAN COMPANY

designing, developing, manufacturing, publishing and distributing microcomputer software

DEALER ENQUIRIES WELCOME
AUTHOR'S SUBMISSIONS INVITED

CALL OR WRITE

(416) 273-6350

PRO·LINE SOFTWARE

755 THE QUEENSWAY EAST, UNIT 8,
MISSISSAUGA, ONTARIO L4Y 4C5

P.O. Box 4364
Flint, Michigan 48504
(313) 233-5731
(313) 233-3125

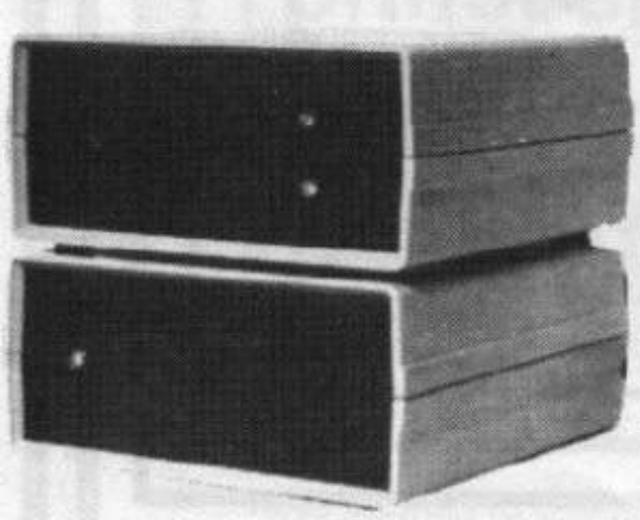
ZANIM SYSTEMS

HOME CONTROL FOR YOUR VIC UNDER $200

The ZCM-1 is the Master Control module that provides the interface between your computer and our line of Zanim Application Modules. Up to 15 Application Modules can be piggy-backed to the ZCM-1 Master Control module. The ZCM-1 is compatable with any standard RS-232 (serial) interface. A special Master Control module, the ZCM-1V is available for the VIC-20 and Commodore-64 computers.

*The ZCM-1V is available for VIC-20 and C-64 users.

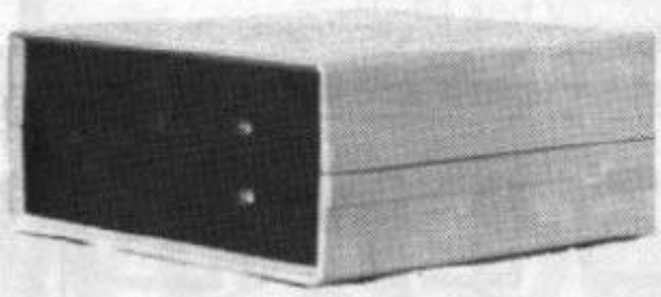
VIC/C64 $129.95
RS-232 $149.95



ZAM-1 is the home control interface module that provides a fully versatile computer controlled environment. ZAM-1 can control up to 256 different lamps and appliances in your home or business providing you with an effective and easy to implement energy management and electric control system. No special modifications are necessary to your building as all control signals are sent over your existing wiring. ZAM-1 can be programmed in BASIC or optional home control software is available. ZAM-1 requires one ZCM-1 Master Control module.

*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-1 Home Control module.

$169.95



The ZAM-2 allows your computer to continuously monitor up to 15 different doors or windows around your home or business. ZAM-2 is a basic building block in a complete computer controlled home security system. With our ZAM-1 Home Control module, you can have a fully integrated security and environment control system. Upon an intrusion, your computer can take the action most appropriate, whether that is to ring an alarm bell, flash all the lights around your home, or dial the police.

*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-2 Security module.

$179.95

The ZAM-3 is a complete telephone answering and dialing system. It is capable of taking the phone off-hook and dialing a number under computer control or of answering the phone when it rings. With the ZAM-1 Home Control module and the ZAM-2 Security module, the ZAM-3 Phone Dialer module can be integrated into a complete home or business security/monitoring system. Applications include security, auto phone dialing, and computer-answering systems.

*Pulse dialing option is available as ZAM-3P. $199.95

*The ZCM-1/ZCM-1V Master Control module is required to use the ZAM-3/ZAM-3P Phone Dialer module.

Prices are in US dollars.

SERIAL OR PARALLEL (CENTRONICS) PORT SWITCHER

DOES YOUR COMPUTER LOOK LIKE THIS?

A PORT SWITCHER NOW AVAILABLE FOR YOUR COMPUTER (ZSW1)

P.O. BOX 4364
Flint, Michigan 48504
(313) 233-5731
(313) 233-3125

☐ Please send me more information or catalogue!

Name _____

Company _____

Address _____

City _____ State _____ Zip _____

What Make/Model Computer do you own?

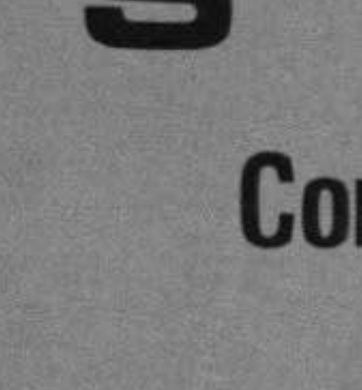
_____

APPLE IS TRADEMARK OF APPLE COMPUTERS, INC.
CPIM IS REGISTERED TRADEMARK OF DIGITAL RESEARCH, INC.
TRS – 80 IS REGISTERED TRADEMARK OF TANDY CORP.
VIC - 20 AND COMMODORE - 64 ARE REGISTERED TRADEMARKS OF COMMODORE, INC.

S O F T W A R E

Commercial Applications For Small Business Computers

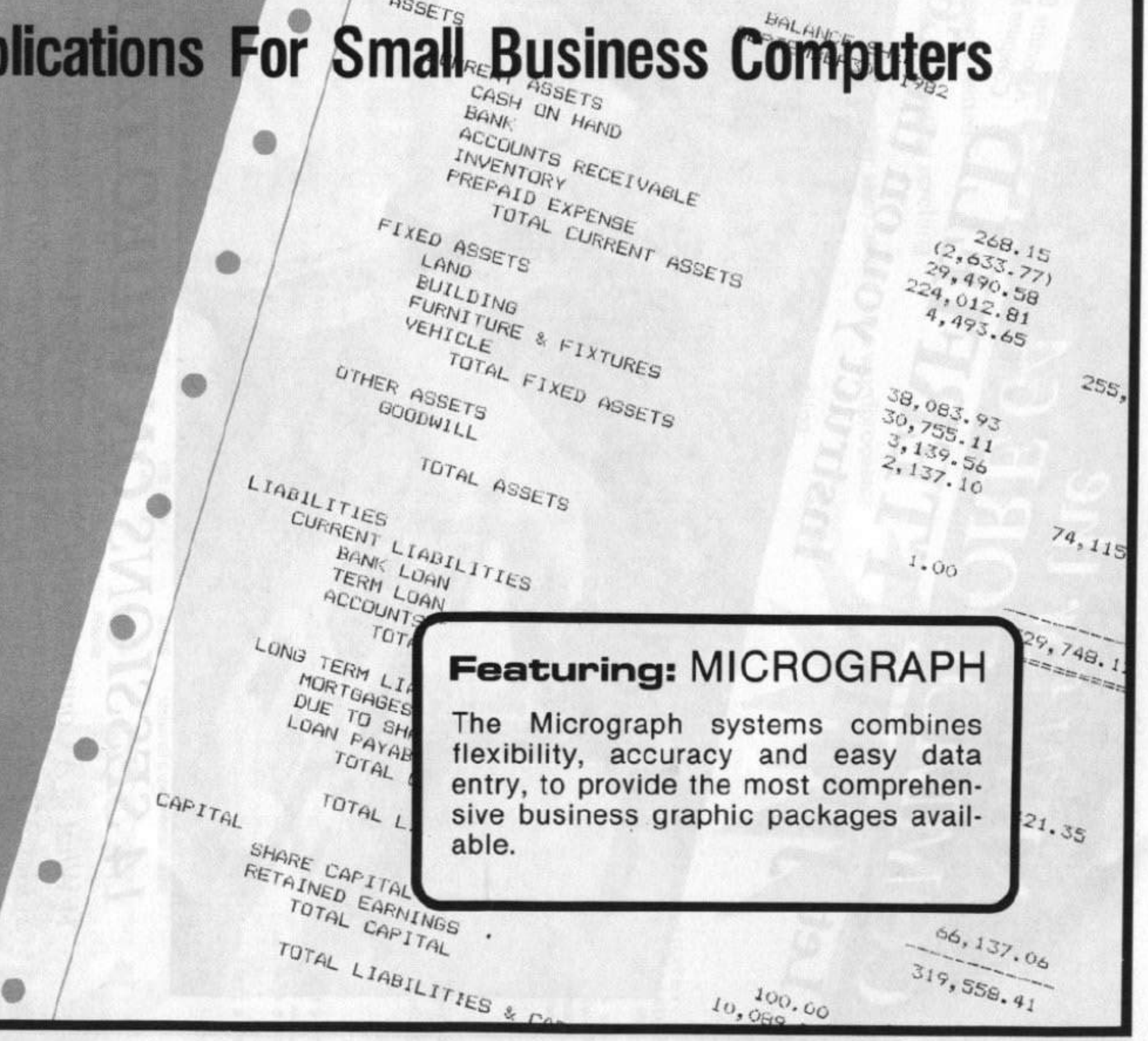
- General Ledger
- Accounts Receivable
- Inventory
- Job Costing
- Payroll
- Property Management
- Micrograph
- Law Office Acct.



BEACON SOFTWARE INC.

705 Progress Avenue, Unit 17,
Scarborough, Ontario M1H 2X1
(416) 431-3200

Formerly BPI Micro Systems Ltd.



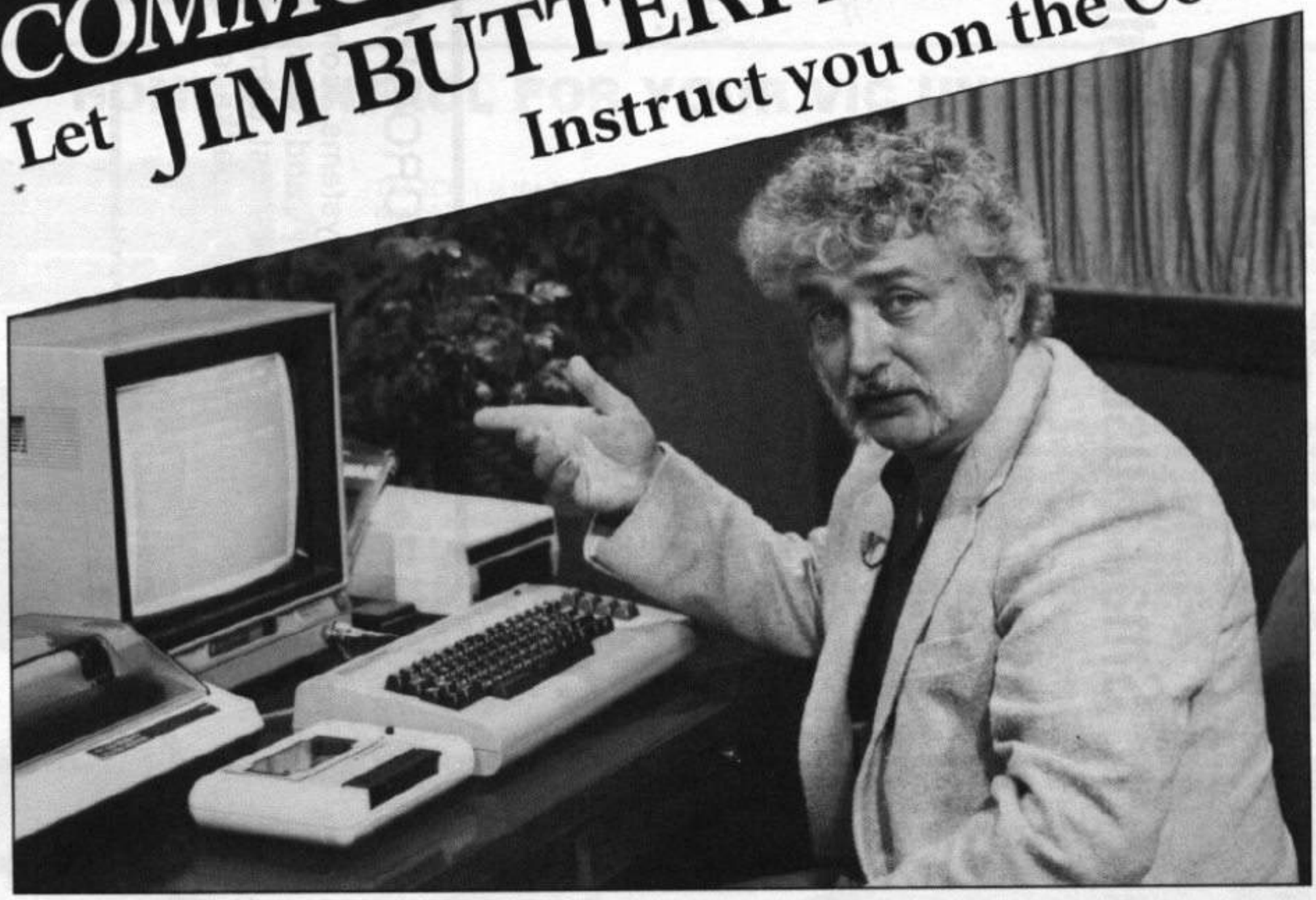
Featuring: MICROGRAPH

The Micrograph systems combines flexibility, accuracy and easy data entry, to provide the most comprehensive business graphic packages available.

NEW for the COMMODORE 64

Let JIM BUTTERFIELD Instruct you on the C64

Associate Editor
Compute Magazine



14 SESSIONS ON VIDEO TAPE

1) What Is A Commodore 64?
2) Getting Started
3) Lets Run Programs
4-A) What Makes Programs Work?
4-B) Putting Programs To Work
5) Storing Information
6) The Commodore 64 As A Learning Tool
7) Computers Talking to Computers
8) Commodore 64 Language
9) Graphics
10) Commodore 64 Working For You
11) Commodore 64 Music
12) Computer Games And Simulations
13) Now What?

(BETA OR V.H.S.)

Order by phone with VISA or MASTER CHARGE
(209) 255-1600

Send $39.95
California residents add 6% sales tax

TO: COMM 64 Training Tape
2727 N. Grove Ind. Drive #101
Fresno, California 93727

Cash, Credit Card, Check, Money Order or C.O.D.
Order by Dec. 10 for Christmas delivery

Look at these Features

- Fully screen-oriented
- Horizontal and vertical scrolling
- Terminal mode — never seen before on a wordprocessor
- Supports Commodore disk and cassette handling
- Imbedded commands

Bliztext

Wordprocessor for Commodore 64

BLIZTEXT is a trademark of ELCOMP PUBLISHING, INC.

Commodore-64 and VIC-20 are trademarks of Commodore Business Machines.

Dealer and Distributor inquiries are invited.

BLIZTEXT — SUPER WORDPROCESSOR for the Commodore-64

— ON SALE NOW! —

- Fully screen-oriented, up/down, left and right scrolling — Upper and lower case
- More than 70 commands
- Full I/O compatibility with Commodore peripherals Upper and lower case
- Works with practically every printer on the market, user definable printer control commands
- INCLUDE command allows handling large files on up to 4 diskettes or on cassette.
- Build in terminal software for electronic mail and networking. Telecommunications mode, upload and download, save on disk or cassette.
- Dynamic formatting, Imbedded commands
- Single keystroke for disk directory and error channel
- Program comes on disk or cassette
- Double line spacing, left and right margin justification, centering, page numbering, and practically everything one expects from a good wordprocessor.

AVAILABLE NOW!

Order # 4965 **$89.00**
Manual only (62 pages) **$29.95**

MACROFIRE
Editor/Assembler for the Commodore-64

ON SALE NOW
AVAILABLE IMMEDIATELY

One outstanding tool, consisting of 3 powerful elements combined into one efficient program!

1.) Fully screen-oriented Editor (more than 70 commands)
2.) Very fast assembler with macro capability
3.) Machine Language Monitor

Assembly can be started from the editor. Translates in 3 passes. More than 1,000 lables, screen oriented/no line numbers, scrolling, includes disk files.
Practically everything the serious machine language programmer needs everyday!

Manual only **$19.95**
Order # 4963 **$89.00**

THE GREAT BOOK OF GAMES, VOL.I,
by Franz Ende
46 programs for the Commodore 64
Introduction to graphics and sound. How to program your own games. Walking pictures, animation, high resolution graphics, programming tips and tricks, hints and useful subroutines for the beginner and advanced programmer. This book is a MUST for every C-64 owner.
Come and get it — It's yours for only **$ 9.95**
Order # 182 **128 pages** **$9.95**
Programs from the book on disk.
Order # 4988 **$19.95**

MORE ON THE SIXTYFOUR, by H.C. Wagner
How to get the most out of your powerful Commodore 64. Very important subroutines, tricks and hints in machine language for your C-64. How to modify DOS. How to connect a parallel and serial printer. How to design your own terminal program for communication and networking. Dig into I/O for cassette and disk.
Order # 183 **$9.95**
Programs from the book on disk
Order # 4989 **$19.95**

NEW PRODUCTS

Watch out for our new books, software and add-ons to come soon. ON SALE NOW! — ORDER TODAY!

How to program in 6502 Machine Language on your C-64 , by S. Roberts (Introduction)
Order-# 184 **$12.95**

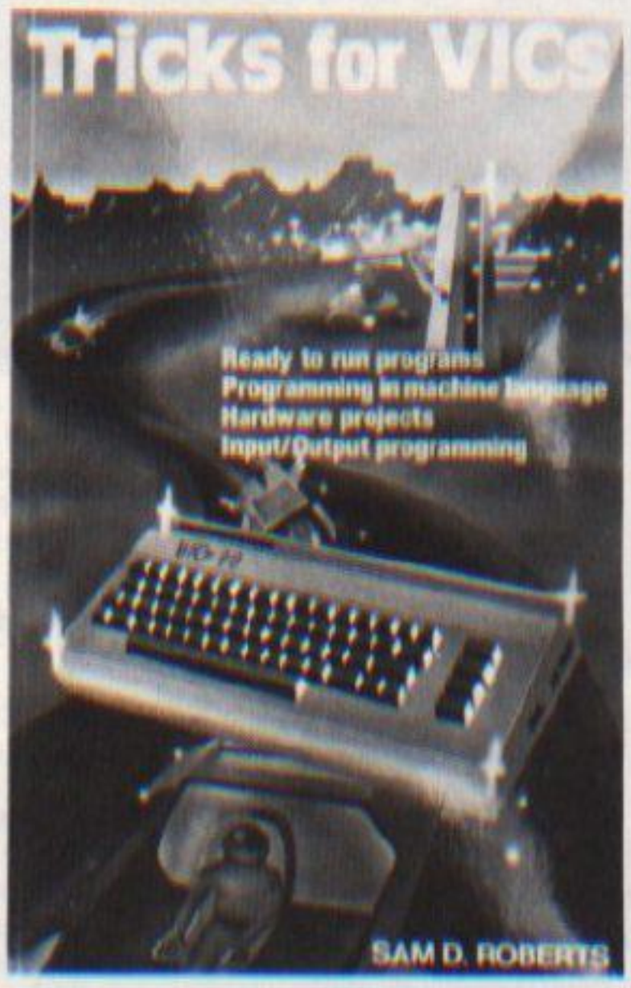
Commodore-64 Tune-up, Vol. I, by S. Roberts
How to expand and customize your C-64.
Order # 185 **$12.95**

Small Business Programs for the Commodore-64
by S. Roberts
How to make money using your C-64. Mailing list, invoice writing, inventory, simple wordprocessing and much more.
Order # 186 **$12.95**

Hardware Add-Ons:

Parallel printer interface KIT	Order # 4990	$ 19.95
Direct Connect Modem KIT	Order # 4991	Ask f.price
Universal Experimenter Board	Order # 4970	$ 9.95
Expansion Board, space for four experimenter boards(board only)	Order # 4992	$ 29.95

For your VIC-20
Tricks for VICs $ 9.95
Universal Experimenter board



HOFACKER

PAYMENT: check, money order, VISA, MASTER CARD, Eurocheck, ACCESS, Interbank
Prepaid orders add $3.50 for shipping (USA)
$5.00 handling for C.O.D.
All orders outside USA: add 15 % shipping, California residents add 6.5 % sales tax.

ELCOMP PUBLISHING, INC
53 Redrock Lane
Pomona, CA 91766
Phone: (714) 623 8314
Telex: 29 81 91

Advertising Index

Software

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Academy Software	VIC20/C64 Software					73	**77**	
Advantage Computer Acc.	VIC 20 games				75			
Beacon Software	Business Packages				74		**85**	
BMB Compuscience	The TOOL (programming aid)	Micro Application					**73**	
	Pro Golf						**76**	
Boston Educational Computing	Educational Software					64		
Cardinal Software	VIC20/C64 Games, Utilities, Edu.						**72**	
Computer Marketing\Canadian Micro	Calc Result (spreadsheet prog.)	Handic Software ab		IBC	IBC	IBC		
Data 20 Corp.	Word Manager					67		
Eastern House	MAE Assembler					68	**81**	
Execom Corp.	40/80 Screen Select					68		
Hofacker	C64 wordprocessor						**87**	
Input Systems Inc.	Typro (wordprocessor)					68	**81**	
Magreeable Software	Stock Helper					66	**75**	
Microcomputer Solutions	C64 Provincial Payroll						**81**	
Micro Ware	C64 JOY-A-WORD						**75**	
	C64 Disk Utility						**82**	
MicroSpec	C64 Business Software						**70**	
Microtechnic Solutions	C64 Terminal software						**77**	
Midwest Micro Inc.	VIC20/C64 Graphics Util					69		
	VIC20/C64 Terminal soft						**83**	
Nüfekop	VIC 20/C64 Games						**1**	
Pacific Coast Software	C64: Utility, Business, Games			62				
Performance Micro Products	C64 Forth					64		
P.F. Communications	J Butterfield video tutor						**86**	
Precision Software	Superscript (wordprocessor)		61					
Pro-Line Software	PAL 64 (assembler)				79	73	**72**	
	POWER 64 (programming aid)				76	69	**74**	
	MailPro					74	**78**	
	general				76	66	**83**	
Psycom Software Int'l	C64 software						**80**	
RAK Electronics	VIC20/C64 Games, Utilities						**80**	
Software International	VIC20/C64 Software					65		
Southern Solutions	Business packages				78	71		
William Robbins Software	VIC/64/PET Software					74	**75**	
Wycor Business Systems	Provincial Payroll			64				

Hardware

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Apropos Technology	VIC20/C64 Printer, Exp board						**79**	
Computer Workshops \Computer Marketing	Z-RAM (CP/M board)	Madison		63				
Connecticut microComputer	PET/CBM Interface adapters			62	79			
	Analog/Digital I/O					64	**78**	
Eastern House	Trap 65					68		
	VIC Rabbit					68		
	Eprom Programmer					68		
	Communications Bd					68		
George M. Drake & Associates	Colour Monitors	Amdek				BC		
Gosub International	Flexikey					75	**82**	
Micro Ware	Tape Interface					70		
	VIC20 RAM Expand					70		
Micro World Electronix	VIC20/C64 Printer Interface					65	**74**	
Midwest Micro Inc.	Smart ASCII Plus					69		
Midwest Peripherals	VIC20 Expander					74	**72**	
Precision Technology	VIC20/C64 Expander Boards			61	79	73		
Richvale Telecommunications	C64 Link (IEEE adapter)			IFC	IFC	IFC		
Zanim Systems	Home control hardware						**84**	

Accessories

Advertiser	Product Name (Description)	Manufacturer	Issue# / Page 1	2	3	4	5	6
Advantage Computer Acc.	Joysticks	Wico			77			
Computer Workshops	Apr83 products list				76			
Consultors Int'l	Stock Market With Your PC (book)		64					
"	A To Z Book Of Games		64					
"	Dynamics Of Money Mgmt (book)		64					
"	Invment. Analysis w/Your Micro (book)		64					
Cursor	C64 Tape Magazine					69	**78**	
Int'l Marketing Services	Disk, printers, misc.						**80**	
Leading Edge Inc.	Elephant Diskettes		BC	BC	BC			
Midnight Software Gazette	Subscriber Info					72	**71**	
Toronto PET Users Group	Membership info			61		75	**74**	

Laser Strike

for the commodore 64

challenge the asteroid field,
maneuver the caves of ice,
experience the thrill,
play laser strike.

Laser strike, written in full machine language for the Commodore 64.

Commodore 64 is a registered trademark of Commodore Business Machines Inc.

Visa/MC/Check/Money Order accepted

In U.S.
Cassette $24.95
Disk $29.95
Isis Hathor Digital Productions
6184 Verdura Ave.
Goleta, CA 93117
(805) 964-6335
Add $2.00 postage and handling
California residents add 6% sales tax

*** Ask about Laser strike posters**

Call Toll Free - 1-800-558-8803

ISIS HATHOR
DIGITAL PRODUCTIONS

In U.K.
Cassette £ 9.00 VAT included
Disk £19.95 VAT included
Isis Hathor U.K.
Andrew Barrow
Royden, Perkslane
Prestwood, Gt. Missenden
Bucks, England HP16 OJD
02406-3224
You will be billed
for postage and handling

COMPATIBLE COLOR-I . . .

NEW 2 YEAR WARRANTY!
On all monitor electronics . . . 3 yrs. on all CRT's
(See details at dealer)

The popular choice for popular computers . . . at a popular price.

The Color-I Monitor is designed to perform superbly with your Apple II, Atari or VIC Commodore personal computer and others. Highly styled cabinet. It accepts a composite video signal to produce vivid, richly colored graphic and sharp text displays. Very reasonably priced, the Color-I is a giant step above home TV sets and other monitors.

Just write, or call to receive complete specifications on the Amdek Color-I Monitor.

- Quality 260(H) x 300(V) line resolution.
- Built-in speaker and audio amplifier.
- Front mounted controls for easy adjustment.
- Interface cables available for Atari and VIC Commodore computers.
- FCC/UL approved.

2201 Lively Blvd. • Elk Grove Village, IL 60007
(312) 364-1180 TLX: 25-4786
REGIONAL OFFICES: Calif. (714) 662-3949 • Texas (817) 498-2334

AMDEK CORP.

Amdek . . . your guide to innovative computing!